Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Зозуля Антон Анатолійович,
студент групи РЗ-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Розробка системи визначення кібератаки підміною OPC-сервера при комп'ютерному управлінні технологічними установками

Спеціальність:
125 Кібербезпека

Керівник:
Стопакевич Олексій Аркадійович,
к.т.н., доцент

Одеса – 2021

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____2021р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Зозулі Антону Анатолійовичу

1.Тема роботи: *Розробка системи визначення кібератаки підміною OPC-сервера при комп'ютерному управлінні технологічними установками*
керівник роботи *Стопакевич Олексій Аркадійович, к. т. н., доцент,*
затверджені наказом ректора від „25" жовтня 2021 р. № 372-в .
2.Зміст роботи*: аналіз проблемної області, постановка задачі, аналіз проблем кібербезпеки АСУТП, моделювання ситуації кіберзлочину шляхом підміни OPC-серверу й розробка алгоритмічного й програмного забезпечення для його подолання*
3. Перелік ілюстративного матеріалу: *слайди презентації.*

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

6. Дата видачі завдання "_____"_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*01.09.2021*	*виконано*
2	*Обґрунтування вибору рішення. Збір даних*	*15.01.2021*	*виконано*
3	*Аналіз основних аспектів кібербезпеки АСУТП*	*01.10.2021*	*виконано*
4	*Розробка програмного забезпечення для моделювання роботи АСУ ТП*	*10.10.2021*	*виконано*
5	*Розробка програмного забезпечення для моделювання кіберзлочину й його виявлення*	*17.10.2021*	*виконано*
6	*Підготовка тексту роботи*	*01.11.2021*	*виконано*
7	*Підготовка презентації та доповіді*	*12.11.2021*	*виконано*
8	*Попередній захист*	*26.11.2021*	*виконано*
9	*Нормоконтроль, рецензування*	*15.12.2021*	*виконано*
10	*Занесення роботи в електронний архів*	*18.12.2021*	*виконано*
11	*Допуск до захисту у завідувача кафедри*	*19.12.2021*	*виконано*

Здобувач вищої освіти _____ *Зозуля А.А.*

Керівник роботи _____ *Стопакевич О.А.*

ЗАВДАННЯ

на розробку розділу “Охорона праці”

Зозулі Антону Анатолійовичу, група РЗ-161

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Тема роботи *Розробка системи визначення кібератаки підміною OPC-сервера при комп’ютерному управлінні технологічними установками*

Зміст розділу:

1. Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.
2. Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях.
3. Дослідження психомоторних показників і оцінка сили нервової системи шляхом експрес-діагностики за методикою теппінг-тесту.

Керівник роботи	Консультант з охорони праці
_____ (О.А.Стопакевич)	_____ (_____)
«____» _____ 2021 р.	«____» _____ 2021 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розробка системи визначення кібератаки підміною OPC-сервера при комп’ютерному управлінні технологічними установками» на здобуття другого (маігстерського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 15 рисунків, 8 таблиць, 6 додатків, 17 літературних джерел за переліком посилань. Робота виконана на 103 сторінках загального тексту і 64 сторінках основного тексту.

Відкрито можливість нової кібератаки на робочі станції керування технологічними установками промислових процесів. Сенс кібератаки полягає в тому, що по мережі, яка зв’язує технологічні станції, робиться підміна OPC-сервера, після чого технологічна установка зловмисником переводиться в аварійний стан таким чином, щоб оператор, спостерігаючи за технологічним процесом за допомогою SCADA-системи не міг помітити розбалансування процесу. Для виявлення кібератаки розроблено спеціальну діагностичну систему. Розроблено відповідне програмне забезпечення та, за його допомогою, всі рішення промодельовано в реальному часі з використанням компонентів реального промислового програмного забезпечення та моделі мережі Modbus. Розроблені рішення та програмне забезпечення можуть бути використані в промисловості.

КІБЕРАТАКА, КІБЕРЗАХИСТ, OPC, OLE FOR PROCESS CONTROL, SCADA, СИСТЕМА КЕРУВАННЯ ТЕХНОЛОГІЧНОЮ УСТАНОВКОЮ.

REFERENCE

Qualification work on "Development of cyber attack detection system by replacing OPC-server in computer control of technological installations" for the second (master's) level of higher education in specialty 125 – Cybersecurity, specialization, educational program: Cybersecurity, contains 15 figures, 8 tables, 6 appendices, 17 literature sources according to the list of references. The work is performed on 103 pages of general text and 64 pages of main text.

This study proposes to pay attention to a vulnerability that has not yet been investigated. This vulnerability is related to the fact that having an extensive network of workstations in the enterprise, an attacker can substitute OPC server on any workstation (usually the most important one in production) with his own OPC server and do it in such a way that on the one hand to bring the technological unit of the selected site to stop or crash, but on the other hand to prevent the operator of technological unit, watching the SCADA-system on the computer screen, from noticing anything. A special diagnostic system has been developed to detect a cyber attack. Appropriate software has been developed and, with its help, all solutions have been simulated in real time using components of real industrial software and Modbus network model. The developed solutions and software can be used in industry.

CYBER-ATTACK, CYBER DEFENSE, OPC, OLE FOR PROCESS CONTROL, SCADA, PROCESS CONTROL SYSTEM.THE WORK CAN BE USED FOR ENHANCING CYBERSECURITY IN THE PROCESS OF INTEGRATED COMPUTER SYSTEMS SOFTWARE DEVELOPMENT.

.

ЗМІСТ

ВСТУП

Відповідно до звіту компанії Positive Technologies [1] за вересень 2021 р. промисловий сектор є однією з основних цілей кіберзлочинців. 91%. Відповідно, сектору промислової кібербезпеки в світі присвячено багато наукових публікацій, але, нажаль, в Україні цією сферою кібербезпеки займаються слабо.

Для розвитку наукової теми підвищення кібербезпеки промислових систем керування була звернута увага на проблему кіберзахисту нижньої ланки керування – операторської станції керування технологічною установкою. В бакалаврській роботі мною було досліджено кіберзахист промислової комп'ютерної мережі нижчого рівня Modbus RTU. Продовжуючи тему, в магістерській роботі буде досліджена проблема кібербезпеки важливого програмного забезпечення – OPC сервера.

Головною метою технології OPC є забезпечення можливості спільної роботи засобів автоматизації, що функціонують на різних апаратних платформах, у різних промислових мережах та вироблених різними фірмами. До поширення технології OPC з'єднання SCADA системи з кожних засобом автоматизації проводилось індивідуально. Існували довгі списки "підтримуваного обладнання", дуже складною була технічна підтримка. При модифікації обладнання потрібно було вносити зміни до всіх драйверів, кожен з яких підтримував протокол обміну тільки з однією клієнтською програмою. Число таких драйверів доходило до сотень. Після появи OPC практично всі SCADA-пакети були перепроектовані як OPC-клієнти, а кожен виробник апаратного забезпечення став постачати свої контролери, модулі вводу-виводу, інтелектуальні давачі та виконавчі пристрої зі стандартним OPC-сервером. Завдяки появі стандартизації інтерфейсу стало можливим підключення будь-якого фізичного пристрою до будь-якої SCADA, якщо вони відповідали стандарту OPC. В той же час додавання в типу структуру системи автоматизації додаткового OPC-серверу призводить до нових загроз в сфері кібербезпеки.

1 ВИЗНАЧЕННЯ ПОТЕНЦІЙНО ВРАЗЛИВИХ ЕЛЕМЕНТІВ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ АСУТП, ФОРМУЛЮВАННЯ СТРАТЕГІЇ КІБЕРАТАКИ І КІБЕРЗАХИСТУ

1.1 Аналіз основних цілей проведення кіберзлочинів в промисловій автоматизації

Загальні втрати компаній від кібератак за оцінками експертів сягають до 10 мільярдів доларів за робочу годину [3]

Відповідно до звіту за вересень 2021 р. компанії Positive Technologies [1] більш ніж 9 з 10 (91%) промислових компанії мають вразливі до кібератак інформаційні системи. Промисловий сектор – це другий по кількістю кібератак сектор в США, й ведучий в розвинутих країнах. Злам комп'ютерної мережі підприємства ще не означає доступу до АСУТП, але й сам по собі може принести компанії значні економічні втрати, пов'язані з втратою важливої інформації, яка представляє комерційну таємницю.

Більшість хакерів як правило задовольняються крадіжкою інформації з метою її перепродажу конкурентам чи за їх попереднім замовленням.

Другою можливою ціллю є вимагання грошей. Як правило в цьому випадку хакери погрожують спрямованим порушенням роботи технологічного процесу чи підприємства у цілому. Аварія на виробництві чи порушення системи автоматизації бізнес-процесів може призвести до значних фінансових втрат компанії, втрати до неї довіри клієнтів, тощо. Для того, щоб не причиняти шкоди хакери вимагають гроші.

Третьою ціллю є знищення певних конкурентів в галузі. В такому випадку хакери себе ніяк не проявляють, а просто спричиняють максимально можливу шкоду підприємству й його репутації.

Четверта ціль – політична. Хакери, які мають політичну мету, можуть як підтримуватись певними державами, так і діяти автономно. Як правило, метою хакерів цієї категорії не є спричинення аварій.

П’ята можлива ціль – терористична. Хакер-терорист може хотіти як вплинути на масову свідомість громадян певної держави, так і просто мати певні особливості психічної активності і уявлень, виходячи з яких буде вважати доцільним спричинення шкоди певному класу людей чи людству у цілому.

1.2 Аналіз деяких найбільш репрезентативних промислових кібератак

Як правило промислові компанії не розповсюджують інформацію про злам чи порушення, однак іноді ситуацію не можливо скрити. Наприклад, влітку 2021 року була проведена атака хакерів Східної Європи на енергосистему США, що призвела до вимкнення системи газопостачання, що забезпечує потреби 45% східного побережжя США [2]. За результатами моделювання ситуації спеціалістами порталу кібер-захисту Standoff за два дні хакери мали можливість спричинити й вибух шляхом різкого перекриття подачі газу.

Також у 2021 р. в США в Флориді була проведена спроба терористичної атаки на станцію очищення води. Метою було збільшення кількості гідроксиду натрію з метою отруєння людей. На щастя атака мабуть робилась не професійним хакером, який використав вразливість в програмі віддаленого доступу TeamViewer. Оператор помітив, що хакер змінив завдання на кількість гідроксиду натрію шляхом переміщення мишею по екрану й атака була майже миттєво нейтралізована.

Інформація, яка наведена в звіті TOP 20 виробничих кібератак [5] представляє значний інтерес для виявлення типових стратегій зламу комп’ютерних й промислових мереж виробничого підприємства. Зазначимо, що за даними Positive Technologies [1] потенційна можливість зламу комп’ютерної мережі промислового підприємства в значній мірі (ймовірність 56%) означає й можливість зламу його АСУ ТП.

Шосте й сьоме місце в TOP 20 серед найбільших виробничих атак займає Україна. Цікаво, що на відміну від перших атак в рейтингу, ці атаки проводили незалежні хакери, а не інсайдери.

Шосте місце займає кібератака хакерів-активістів 2016 р. на компанії-розподілювачі електроенергії України. Хакери провели викрадення паролю до віддаленого доступу шляхом простого фішингу. Після отримання паролю хакери зломали контролер домену Windows, створили для себе акаунти з максимальними правами доступу й отримали доступ до АСУ ТП. Після вивчення параметрів й аналізу екранів людино-машинних інтерфейсів хакери спричинили аварію, видалили всю інформацію з жорстких дисків й перепрошили все промислове обладнання до якого мали віддалений доступ. Це призвело до відімкнення електроенергії для 200 тис. споживачів протягом 8 годин. Пошкодження змогли доволі швидко подолати лише по причині того, що хакери погано розуміли автоматизацію, не розуміли принципу роботи систем автоматичного керування й алгоритмів розподілення електроенергії. Подолання атаки пройшло в основному фізичним шляхом: на кожній виведеній з ладу підстанції керуючі комп'ютери були відімкненні й електроенергія була направлена в необхідному напряму шляхом ручного управління.

1.3 Короткий огляд програмного забезпечення АСУТП

В виробничих АСУТП системи як правило будуються по трьохрівневому принципу.

Верхній рівень – це рівень візуалізації, диспетчеризації (моніторингу) й збору даних.

Середній рівень – це рівень для з'єднання верхнього рівня з нижнім. На ньому реалізуються промислові мережі, працюють програмовані логічні контролери, системи протиаварійного захисту, тощо.

Нижній рівень – це польовий рівень АСУТП, на якому працюють різні давачі і виконавчі механізми.

Програмне забезпечення для різних рівнів реалізується дуже по-різному.

Для нижнього рівня це, як правило, певні прошивки для флеш-пам'яті, які дозволяють калібрувати і модифікувати алгоритми інтелектуальних давачів і виконавчих механізмів.

Для середнього рівня це програмне забезпечення представляє собою програми для програмованих логічних контролерів і контролерів аварійного захисту. Як правило мови програмування таких контролерів дуже спрощені, тому не можуть використовуватись для реалізації складних алгоритмів керування (в т. ч. автоматичних регуляторів).

Для верхнього рівня програмне забезпечення представляє собою програми для персональних комп'ютерів. Це спеціалізоване програмне забезпечення, яке забезпечує зворотній зв'язок між оператором і елементами нижніх рівнів АСУТП. Обов'язковим функціоналом, який має бути реалізовано на верхньому рівні, є людино-машинний інтерфейс, який призначено для спостерігання за ходом технологічного процесу оператором. Але, частіш за все функціонал програмного забезпечення верхнього рівня значно ширший: це не тільки нагляд, але й втручання в хід технологічного процесу, запис і аналіз технологічних параметрів, зв'язок з іншими інформаційними системами для корекції поведінки процесу в режимі он-лайн, тощо. Типовим програмним забезпеченням, за допомогою якого можливо реалізувати все перелічене, є SCADA-система.

1.4 Архітектура сучасних SCADA-систем

SCADA – це комп'ютерна система, яка призначена для керування й моніторингу технологічних процесів. В широкому сенсі SCADA-система розглядається як програмне й комп'ютерне апаратне забезпечення АСУТП. В більш вузькому, в якому й будемо розуміти цей термін далі, як спеціалізоване програмне забезпечення з певним функціоналом, яке класифікується виробником як SCADA-система. В теперішній час мінімальний функціонал SCADA-системи включає: збір й передача технологічних параметрів, реалізація людино-машинного інтерфейсу, система повідомлень й тривог (алармів), засоби програмування (релейно-контактні схеми, блокове програмування, скрипти, повноцінні мови програмування). Будь-яка SCADA-система має два режими: режим проєктування й режим виконання проєкту й подібна до сучасних RAD-засобів програмування, таких як Visual Studio, Delphi та ін.

Сучасні SCADA-системи, які використовуються для керування промисловими технологічними процесами, реалізуються за клієнт-серверною технологією.

При клієнт-серверному підході безпосередньо з засобами автоматизації проводить комунікацію сервер SCADA-системи. Сервер підключається до промислової мережі й відокремлених засобів автоматизації за допомогою спеціальних систем отримання даних. Системи отримання даних можуть бути реалізовані на програмованих логічних контролерах (ПЛК), спеціальних платах аналого-цифрового перетворення та ін. З появою промислових давачів й виконавчих механізмів, які мають не аналогове з'єднання, а підключаються безпосередньо до промислової мережі, необхідність застосування ПЛК чи спеціальних плат підключення в загальному випадку відсутня, хоча вибір на їх користь може бути просто наявності вже працюючого рішення, яке проєктувальник не має бажання модернізувати без вагомих причин. Більш доцільним є застосування в якості системи даних спеціального сервера інформаційних параметрів. В промисловості як правило використовуються сервери OPC/OPC UA, в більш простих застосунках (домашня автоматизація) – сервери MQTT. Більш докладно питання різниці технологій для побудови серверів інформаційних параметрів буде розглянуто в наступному підрозділи.

Типовий сервер SCADA-системи виконує наступні функції:

- підключається до OPC-серверів, періодично зчитує й проводить запис тегів;
- виконує скрипти;
- працює з серверами керування базами даних;
- проводить авторизацію SCADA-клієнтів та взаємодіє з ними за допомогою спеціального протоколу.

Клієнти SCADA-системи виконують лише задачі візуалізації, всі необхідні дані запитуються у сервера SCADA-системи.

Таким чином, при використанні клієнт-серверної архітектури досягаються наступні цілі:

- суттєво знижується навантаження на комп'ютерну та промислову мережу;

- спрощується процедура супроводження SCADA-системи, так як при зміні проєкту не потрібно його вручну розмножувати на всіх клієнтських комп'ютерах;

- знижується загальна вартість системи.

1.5 Сервер інформаційних параметрів промислової АСУТП

OPC – це програмна технологія, яка представляє собою єдиний інтерфейс для керування різними приладами та обміну даними. Реалізувавши підтримку OPC-клієнта, розробники SCADA-систем позбавились від необхідності підтримувати сотні драйверів для будь-яких пристроїв. Технологія OPC включає ряд стандартів, які описують набор функцій певного призначення. Поточні поширені стандарти:

OPC DA (Data Access) — найбільш поширений стандарт. Описує набор функцій обміну даними в реальному часі з пристроями.

OPC HDA (Historical Data Access) – надає доступ до вже збережених даних і історії.

OPC AE (Alarms & Events) — надає функції повідомлення за вимогою про певні події: аварійні ситуації, дії оператора та ін.

Крім того, існують і інші стандарти, які не набули популярності.

OPC Batch — надає функції рецептурного керування.

OPC DX (Data eXchange) — функції для обміну даними між OPC-серверу.

OPC Security — організація прав доступу клієнтів до OPC-серверу.

OPC XML-DA (XML-Data Access) — описує набор функції обміну даними за допомогою технологій XML, SOAP і HTTP. Його недолік – доволі повільний.

OPC Complex Data – додаткова специфікація до OPC DA і XML-DA, якак дозволяє працювати зі складними типами даних, такі як бінарні структури і XML-документи.

Найпоширенішою з технологій є стандарт OPC DA. Його основний недолік – він побудований на технологіях операційної системи Windows OLE, ActiveX, COM/DCOM. Спроби реалізації її на других системах мовою Java виявились доволі ненадійними. Зараз існує альтернативне рішення для операційних систем, які не належать до Windows – стандарт OPC UA (Unified Architecture), який використовує технологію спрощених web-сервісів (не SOAP, а JSON). Але якщо використовуються лише операційні системи Windows в АСУТП, застосування цього стандарту не дуже доцільно, він ще менше задовольняє вимогам реального часу, а відсутність стандартної бібліотеки Microsoft дозволяє розробникам розробляти OPC-сервери з несумісними реалізаціями стандарту OPC UA.

Як правило, технологію OPC застосовують для обміну даними між польовими пристроями і контролерами і SCADA-системою. Також є можливою організація складних систем на різних рівнях АСУТП, але на це в даному дослідженні звертати увагу не будемо.

Програмне забезпечення (ПЗ) OPC складається з двох частин: OPC-клієнту і OPC-серверу. Програмне забезпечення OPC-серверу через драйвери пристроїв за польовими шинами опитує різні пристрої. ПЗ OPC-клієнту як правило вбудовано в SCADA-систему та призначено для отримання і передачі даних з OPC серверу.

OPC DA сервер забезпечує обмін даними (запис і читання) між клієнтською програмою (звичайно SCADA-системою) та кінцевими пристроями. Дані в OPC представляють собою структуровану змінну (тег) з певними властивостями. Змінна може бути будь-якого типу, який є припустим в OLE: будь-які цілі та дійсні числа, логічний тип, строковий, дата, масив та ін. Властивості можуть бути обов’язковими, рекомендованими і користувацькими.

Обов’язкові властивості це: поточне значення змінної, її тип і права доступу (читання чи/або запис), мітка часу, якість замінної. Якість змінної залежить від виходу вимірюваної змінної за межі динамічного діапазону, відсутності даних, помилок в каналі зв’язку та ін. Як правило якість оцінюється

сервером за довільними алгоритмами за наступною шкалою: добра, погана, невизначена, додаткова інформація.

Додатково можуть бути вказані необов'язкові властивості: діапазон вимірювання значення, одиниця вимірювання, текстовий опис та інші параметри, які визначає користувач.

Для читання даних з OPC-серверу можуть бути використані різні режими.

Синхронний режим. Клієнт посилає запит серверу й чекає відповіді від нього.

Асинхронний режим. Клієнт відправляє запис і переходить зразу ж до виконання інших задач. Сервер після обробки запиту посилає клієнт повідомлення про готовність даних і клієнт їх забирає.

Режим підписки. Сервер відсилає клієнт тільки ті теги, які змінились. Для того, щоб шум даних не був прийнятий за їх зміну вводиться поняття «мертвої зони», яка дещо перевищує максимально можливий розмах похибки.

Режим оновлення даних. Клієнт викликає одночасне читання всіх активних тегів. Активними називаються ті теги, які не позначені як пасивні. Таке ділення може бути використане для спрощення навантаження на сервер, якщо режим підписки витрачає забагато ресурсів.

Тести показують, що SCADA-система працює приблизно в 2 рази швидке при роботі напряму з промисловими протоколами, ніж через проміжний OPC-сервер [7]. Однак OPC-сервер став типовою промисловою практикою й обов'язковою вимогою багатьох технічних завдань на АСУТП по причині можливості окремого відлагодження промислової мережі і програмного забезпечення АСУТП. Вартість деяких OPC-серверів досягає 1000 доларів за ліцензію на 1 комп'ютер, однак це оправдує себе можливістю реалізувати віртуальну мережу для відлагодження SCADA-системи. З іншого боку, широкі можливості OPC-серверу додають і загрози кібербезпеки АСУТП, про що буде в наступних підрозділах.

1.6 Основні методи захисту промислової системи автоматизації

Приведемо схему захисту й рекомендовані практики, які наведені в звіті [5].

Перелічимо «найкращі практики» захисту промислових систем автоматизації (СА):

1) Проводити сегментацію мереж за допомогою файрвалів;
2) Проводити зашифровану передачу даних (в разі можливості);
3) Встановити індивідуальні паролі на всі пристрої, інформаційні системи, програмне забезпечення і т.п. до якого можливо мати віддалений доступ;
4) Використовувати антивірусне програмне забезпечення;
5) Пропускати увесь трафік на рівні маршрутизаторів через систему аналізу трафіку;
6) Регулярно оновлювати програмне забезпечення в разі знаходження в ньому вразливостей.

Більш докладний аналіз практик безпеки проведено в стандарті ISA/IEC 62443.

Відмітимо, що сегментація мереж є однією з дуже розповсюджених рекомендацій. Це є не тільки рекомендаціями стандартів (наприклад, стандартів ISA), а й промисловою практикою, яка впроваджується передовими компаніями-постачальників систем автоматизації. Однак повна сегментація і демілітарізована зона, виходячи з особливостей і складності протоколів систем автоматизації, доменної системи Windows й хмарних рішень, як правило не досяжна. Тому ця, як і інші практики, не досягають часто повної безпеки чи вимагають багато грошей й роблять модернізацію системи автоматизації (навіть прості оновлення безпеки з невеликою зміною протоколу) дуже складною.

1.7 Аналіз результатів досліджень рівня захищеності працюючих АСУТП

Найбільш репрезентативним є результати звіту CyberX. Автори звіту провели дослідження 1821 промислової мережі за допомогою спеціального авторського програмного забезпечення, яке дозволяє виявити типові помилки

конфігурації і відмові вразливості програмного забезпечення приладів та систем автоматизації. Особливістю дослідження є те, що воно не присвячено певні галузі, а покриває промислові АСУТП різного призначення: роботехнічні, хімічні, холодильні, енергетичні, нафто і газо-переробні, траспортувальні, шахтові. Також були досліджені і системи домашньої автоматизації.

Основні результати досліджень наступні:

- значна кількість комп'ютерів працює під керування операційної системи Windows з версією, яка вже не підтримується фірмою Microsoft (7, XP, 2000...). До припинення підтримки ОС 7 така кількість складала 62%, після – 71%;

- більшість (64%) систем віддаленого доступу використовують незашифровані паролі, які можливо відслідкувати сніффером;

- 22% працюючих систем інфіковані чи були інфіковані вірусами (лишились сліди їх активності).

Також цікаве дослідження Лабораторії Касперського [9]. В ньому представлена ступінь відносної небезпеки кожного з компонентів промислової АСУТП.

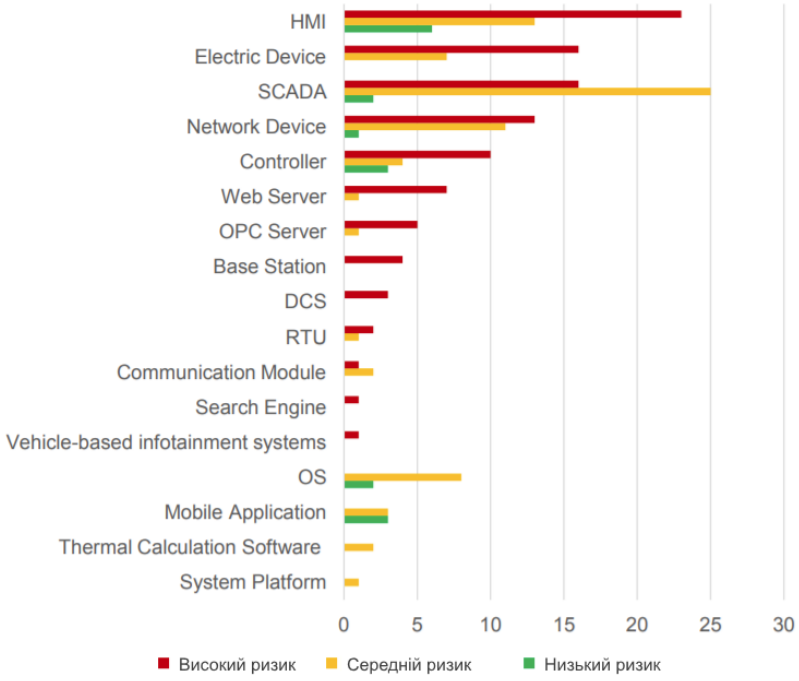


Рисунок 1 – Відносний ризик елементів промислової АСУТП [9]

З рис. 1 бачимо, що доволі часто ціллю атаки є атака на SCADA, HMI чи контролер.

Однак слід відміти, що дещо складними, але значно менш помітними, мають бути атаки на OPC-сервер. Ця тема не дуже розглядається в світовій літературі, однак атака на OPC-сервер може бути значно більш ефективною при професійному підході, ніж атака на SCADA, що як правило є більш помітним.

У 2020 році команда дослідників Claroty проаналізувала продукти кількох фірми-поставників програмних засобів АСУТП [10], які використовують сімейство протоколів Open Platform Communications (OPC DA, AE, HDA, XML DA, DX та OPC UA). Перевірені програмні засоби використовуються як сторонні рішення багатьма великими виробниками систем промислової автоматизації, такими як Rockwell Automation та GE. Результати аналізу виявилися невтішними : рішення на основі даних бібліотек схильні до множинних уразливостей, які можуть призвести до відмов обладнання, віддаленого виконання коду та витоку критичних даних.

Розглянемо знайдені вразливості в OPC-серверах відомих фірм [10].

1) Softing Industrial Automation GmbH

CVE-2020-14524 : переповнення буфера на основі купи (CWE-122)

CVE-2020-14522 : неконтрольоване споживання ресурсів (CWE-400)

Softing Industrial Automation GmbH - постачальник рішень для моніторингу та діагностики комунікаційних мереж. OPC Software Platform від Softing Industrial Automation є рішенням, що забезпечує взаємодію між OPC UA та OPC Classic, а також реалізує підключення до хмарних ресурсів. Інтегрований сервер OPC UA Server надає доступ до даних з ПЛК Siemens, Rockwell Automation, B&R Industrial Automation, Mitsubishi та іншим ПЛК, що підтримують обмін даними протоколу Modbus.

Експлуатація виявлених уразливостей не вимагає від атакуючої наявності особливих технічних навичок, уразливості піддаються всі версії Softing молодше 4.47.0.

Першу вразливість (CVE-2020-14524) було виявлено у бібліотеці OPC DA XML Softing HTTP SOAP-сервера. Веб-сервер не обмежує довжину та не санатизує (екранує) значення у заголовку SOAP-повідомлень. Використання надто великих заголовків SOAP зрештою призводить до вичерпання ресурсів динамічної пам'яті. Оскільки веб-сервер не перевіряє результат операції виділення пам'яті, дані будуть записані в неініціалізовану область, що призведе до збою веб-сервера. Цій критичній вразливості надано рейтинг 9.8 згідно з CVSS v3.1.

Друга вразливість (CVE-2020-14522) полягає у використанні неприпустимого значення для деяких параметрів, що викликає нескінченний цикл виділення пам'яті, і в результаті надмірне споживання пам'яті та відмова в обслуговуванні.

2) Kepware PTC

CVE-2020-27265 : переповнення буфера на основі стека (CWE-121)

CVE-2020-27263 : переповнення буфера на основі купи (CWE-122)

CVE-2020-27267 : безкоштовне використання (CWE-416)

Kepware є компанією-розробником комунікаційного програмного забезпечення для підприємств промислового виробництва, нафтогазової галузі, електроенергетики та систем автоматизації будівель. Рішення Kepware використовуються в SCADA-системах для з'єднання з промисловими пристроями Allen Bradley, AutomationDirect, BACnet, DNP 3.0, GE, Honeywell, Mitsubishi, Modicon, Omron, Siemens, Texas Instruments, Yokogawa та ін.

Вразливими є версії:

KEPServerEX v6.0 - v6.9;

ThingWorx Kepware Server v6.8 - v6.9;

ThingWorx Industrial Connectivity - всі версії;

ThingWorx OPC-Aggregator - всі версії.

У ThingWorx Edge Server виявлено вразливість переповнення стека (CVE-2020-27265), яка не вимагає аутентифікації та може експлуатуватися віддалено.

Помилка в логіці декодування рядків OPC дозволяє записувати рядки довші за 1024 байт без виділення додаткової пам'яті. Експлуатуючи вразливість, зловмисник може перезаписати дані у стеку після перших 1024 байт і викликати збій у роботі сервера та, можливо, здійснити виконання шкідливого коду (рейтинг CVSS v. 3.1 - 9.8).

Ще одна вразливість потоку декодування рядків OPC (CVE-2020-27263) може призвести до витоку даних та відмови сервера через читання поза межами виділеної динамічної пам'яті. Вона є як у Windows-, так і Linux-версіях ThingWorx Edge Server (рейтинг CVSS v. 3.1 — 9.1).

У Kepware KEPServerEX Edge Server є можливість використання даних після звільнення пам'яті. Експлуатація вразливості (CVE-2020-27267) не вимагає аутентифікації та викликає «перегонку потоків», що призводить до спроби використання звільненого об'єкта після закриття з'єднання з ним та наступного збою сервера (рейтинг CVSS v. 3.1 - 9.1).

3) Тунелі Matricon Honeywell OPC UA

CVE-2020-27297 : переповнення купи через переповнення цілого числа (CWE-122)

CVE-2020-27299 : витік інформації через зчитування OOB (CWE-125)

CVE-2020-27274 : неправильна перевірка на наявність незвичайних або виняткових умов (CWE-754)

CVE-2020-27295 : неконтрольоване споживання ресурсів (CWE-400)

Matrikon — компанія-постачальник програмних рішень з промислової автоматизації для ABB, Honeywell, GE, IBM, Oracle, Rockwell Automation, Schneider Electric, Shell, Siemens, Wonderware та ін. серверами OPC UA, але й із серверами та клієнтами OPC Classic.

Вразливими є всі версії Matrikon OPC UA Tunneller молодше 6.3.0.8233.

У компонентах Matrikon OPC UA Tunneller були знайдені множинні вразливості, включаючи критичне переповнення динамічної пам'яті (купи) (CVE-2020-27297, рейтинг CVSS v. 3.1 - 9.8) та витік даних через читання області пам'яті за межами купи (CVE-2 -27299).

Експлуатація даних уразливостей може призвести до збою в роботі сервера або контролю області пам'яті поза буфером і виконання шкідливого коду. Тобто зловмисник може отримати контроль над OPC-сервером для подальшого просування у мережі.

Раніше, в 2018 році, Kaspersky ICS CERT випустив дослідження безпеки протоколу OPC UA [11] , де повідомляється про 17 знайдених і вже закритих вразливостей.

Характер виявлених вразливостей Claroty свідчить, що з моменту виходу звіту Kaspersky ICS CERT залишається актуальною проблема контролю якості коду функцій стека протоколів OPC. А отже OPC-сервер залишається одним з вразливих елементів сучасної АСУТП.

1.8 Розробка стратегії кібератаки на OPC-сервер

Для атаки необхідно провести два зламу:

1) Отримати доступ до комп’ютера в мережі і за необхідності підвищити свої права доступу на ньому. Доступ реалізується через інтернет.

2) Отримати доступ через локальну мережу підприємства до комп’ютера з OPC-сервером.

Якщо злами пройшли успішно, то хакер має перейти в режим спостереження за технологічними параметрами, які поступають і отримуються з OPC-серверу SCADA-системою і передаються до промислової мережі. Будемо вважати, що хакер не має інсайдера, тому не може отримати доступ до SCADA-системи, не має моделі технологічного процесу і налаштувань регуляторів.

Приймемо, що метою хакера є спричинення аварії на виробництві, тому він має провести непомітно підміну OPC-серверу таким чином, щоб SCADA-система та інші інформаційні системи підприємства нічого не помітили під час

виведення технологічного процесу в неконтрольований стан. Після цього, звільнивши доступ OPC-серверу до промислової мережі хакер має вияснити положення яких виконавчих механізмі треба змінити і яким чином виключити систему аварійного захисту (за наявності його на польовому рівні, а не, наприклад, в SCADA).

Принципова схема кібератаки показана на рис. 2.

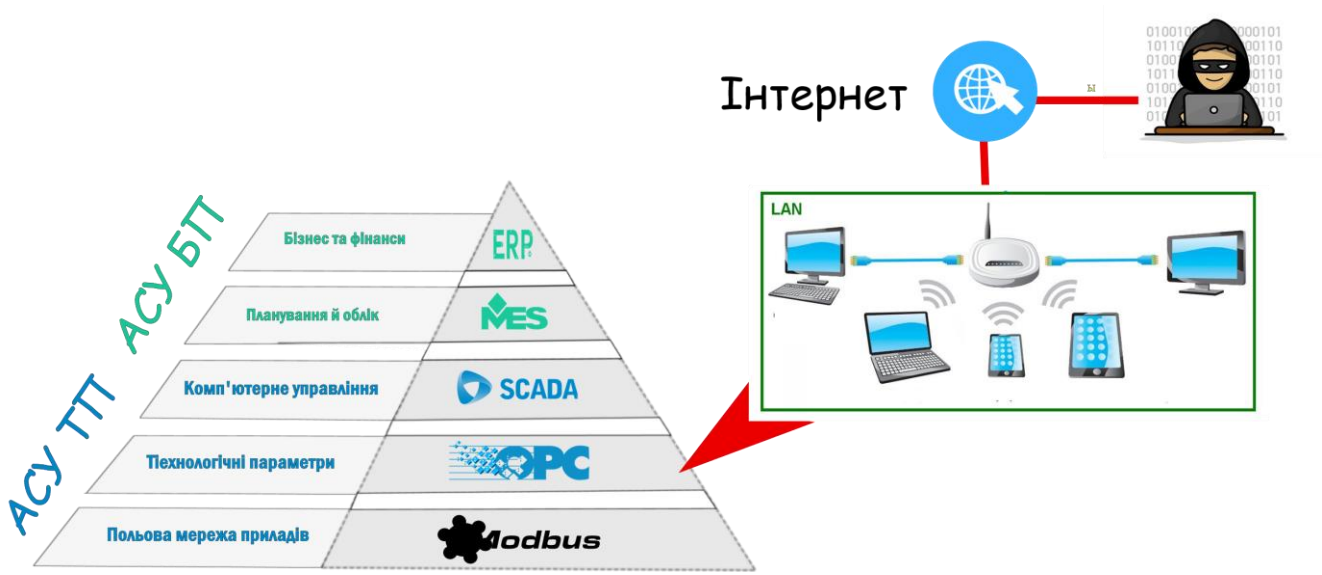


Рисунок 2– Принципова схема кібератаки

Схему захисту від такого роду атак розглянемо в наступному підрозділі.

1.9 Розробка стратегії кіберзахисту АСУ ТП від атаки на OPC-сервер

В першу чергу оператор має постійно слідкувати за оперативними параметрами, якщо вони різко змінилися, то це може бути кібератака.

Далі, якщо хакер поставив лише постійні нережимні параметри (продовжив їх), то така атака буде помітною по ненормальній роботі автоматичних регуляторів, які будуть постійно змінювати положення, оскільки немає реакції керованої змінної.

Третє, в випадку, якщо хакер реалізував плавне і реалістичне продовження трендів керованих змінних за допомогою реалізації програмного моделювання в OPC-сервері таким чином, що автоматичні регулятори не виходять особливо за межі нормальної роботи, людина все не зможе виявити підміну. В такому

випадку в SCADA-системі необхідно реалізувати систему захисту, яка перевіряє особливості роботи певного обладнання, яке хакер наряд буде відтворювати в імітаційній моделі. Це доволі складна робота, яка вимагає забагато часу, вивчення технічної документації всіх засобів автоматизації, тощо. Не завжди її можливо і взагалі якісно реалізувати не будучи інсайдером.

Висновки за розділом

1. Захист промислових АСУТП є актуальною і не дуже розвинутою науковою проблемою

2. Аналіз цілей хакерів та реальних фактів зламу показує, що промисловий сектор є один з найбільшою ціллю кібератак в розвинутих країнах.

3. При розробці програмного забезпечення АСУТП як правило не залучались спеціалісти з кібербезпеки, тому таке програмне забезпечення потенційно має значну кількість уразливостей, значну кількість яких знаходять кожен рік за допомогою нескладних інструментів (фаззінг та ін.), якими володіє будь-який хакер.

4. Як правило розглядають питання захисту контролерів чи SCADA. В цій роботи запропоновано розглянути стратегію кібератаки атаки, яка може використосувати лише OPC-сервер, який теж є доволі вразливим елементом. При якісному проведенні такої атаки оператор нічого не помітить до того часу, доки технологічний об’єкт не буде виведений в аварійний режим роботи.

2 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ ІМІТАЦІЙНОЇ КОМП'ЮТЕРНОЇ МОДЕЛІ АСУТП

2.1 Визначення необхідних компонентів для реалізації імітаційної комп'ютерної моделі АСУТП

Комп'ютерна модель має складатись з програмного забезпечення, яке використовується на виробництві для керування технологічними процесами. Це SCADA-система, OPC-сервер, система для імітації нуль-модемного з'єднання. Крім того, необхідно провести вибір мови програмування та необхідних бібліотек для розробки програмного емулятора технологічного процесу.

2.2 Вибір SCADA-системи

В якості SCADA виберемо Simple-SCADA. ЇЇ ключові переваги для дослідження:

- Демонстраційна версія (на 2 години) безкоштовна;
- Орієнтована на роботу з OPC-сервером (OPC DA/UA). Немає зайвих інтерфейсів доступу;
- Для написання скриптів використовується повноцінна мова програмування Free Pascal, перед виконанням скрипти компілюються;
- Клієнт-серверна архітектура;
- Зручна в роботі і доволі надійна з точки зору зберігання інформації;
- Інтерфейс автоматично масштабується під різні екрани, завдяки використанню відео-режиму
- Інформація про наявність злому за більше ніж 5 років існування SCADA в інтернеті не поширювалась

Приведемо більш докладний опис цієї системи.

Simple-Scada – призначена для розробки та забезпечення роботи в реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або управління. Головна мета проекту – це простота та зручність для кінцевого користувача.

Simple-Scada може вирішувати такі завдання:

- Обмін даними з промисловими контролерами та платами введення-виведення в реальному часі за допомогою технології OPC.
- Обробка інформації та управління процесом у реальному часі.
- Відображення інформації на екрані монітора у зручній та зрозумілій для людини формі.
- Ведення бази даних реального часу із технологічною інформацією.
- Аварійна сигналізація та керування тривожними повідомленнями.

Simple-Scada включає наступні компоненти:

- OPC-клієнт, що забезпечує зв'язок Scada-системи з промисловими контролерами та іншими пристроями введення-виведення інформації.
- Систему реального часу, що забезпечує обробку даних у межах заданого тимчасового циклу з урахуванням пріоритетів.
- Програму-редактор для розробки людино-машинного інтерфейсу.
- Базу даних реального часу, що забезпечує збереження історії процесу в режимі реального часу.
- Систему управління тривогами що забезпечує автоматичний контроль технологічних подій, віднесення їх до категорії нормальних, запобіжних чи аварійних, а також обробку подій оператором чи комп'ютером.

Остаточно, надамо короткий опис мови програмування Free Pascal, яка використовується в Simple-Scada.

Мова Free Pascal й однойменний компілятор зараз мають такі переваги:

- Принципово повнофункціональна мова з підтримкою функціонального програмування і ООП, викликів до бібліотек, має також можливість застосування асемблерних вставок;
- Наявність вбудованих бібліотек для швидкої роботи з популярними СКБД;
- Компілятор безкоштовний;
- Можливість компіляції програми без зміни програмного коду під різні платформи (не ідеально, але якщо з текстових інтерфейсом, то це правда);

- Не потрібні файли зборки, компілятор автоматично визначає що потрібно перекомпілювати;
- Швидкість скомпільованих програм дуже висока;
- Розмір оперативної пам’яті для виконання програми порівняно з багатьма мовами програмування порівняно невеликий;
- Швидкість компіляції не зростає експоненційно з розміром програми, як в деяких інших компіляторах.

Мова Free Pascal, як й інші варіанти Object Pascal має й зворотну сторону – вона не дуже зручна для написання програм з алгоритмами, які не мають чіткого математичного запису. Необхідність визначення всіх змінних, строга типізація, збільшують час написання простих програм.

Проте в багатьох програмних засобах – як для ПЛК, так і для SCADA мови програмування, подібні до Pascal, є типовим вибором. Наприклад, можливо зазначити мову Structured Text, яка дуже поширена як в середовищах програмування ПЛК, так і в SCADA. Таким чином, мова Pascal є традиційним вибором для цієї галузі.

2.3 Вибір OPC-серверу

В якості OPC-серверу виберемо МПС Софт Modbus Universal MasterOPC Server. Його ключові переваги для дослідження:

- До 32 тегів включно безкоштовний;
- Повністю реалізує в нативному режимі протоколи OPC DA і OPC HDA;
- Зручний інтерфейс для налаштування і перегляду роботи серверу;
- Вбудована мова програмування Lua для написання скриптів;
- Підтримка різних особливостей реалізація протоколу ModBus (порядок байтів, запис і читання чисел одинарної і подвійної точності, тощо).

Сервер постійно розвивається і адаптується до різних приладів. Відмітимо підтримку різних польових мереж різних виробників (Овен, IP CAS, Delta Electronics та ін.

Існує більш продвинута версія цього серверу – мульти-протокольна, яку ми застосовувати не будемо . Ця версія серверу рекомендується для зв’язку між OPC-серверами через інтернет. OPC UA та інші подібні протоколи (наприклад, MQTT) реалізуються за допомогою спеціальних плагінів-перетворювачів. Приклад такого застосування показаний на рис. 3 [13]

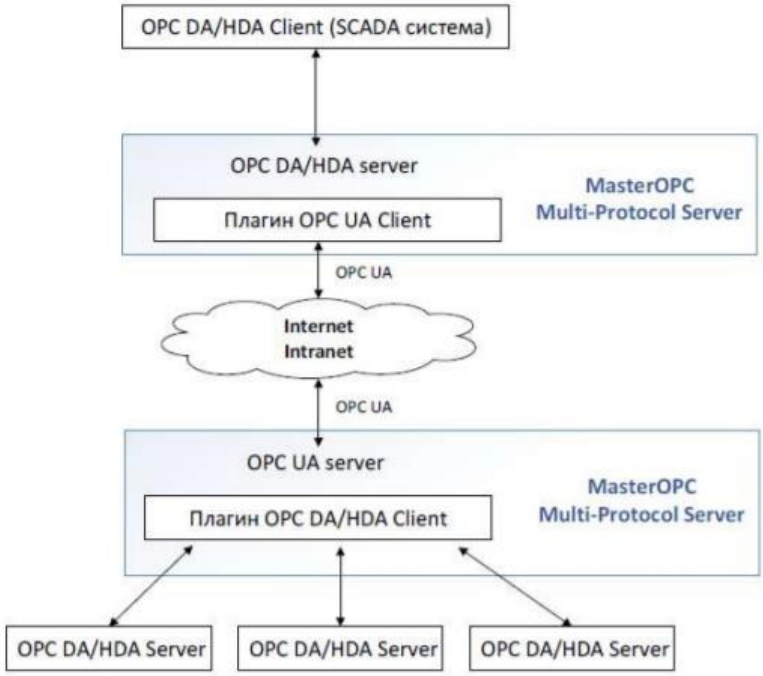


Рисунок 3 – Приклад застосування мульти-протокольної версії OPC сервера

В якості мови програмування в MasterOPC використовується мова Lua. Наявність мови програмування – це основна відмінність професійного OPC-серверу. Lua – динамічна мова і мультипарадигмальна мова. Вона підтримує функціональний стиль програмування та ООП. В результаті вона підходить не тільки для вирішення якихось найпростіших завдань, але і для більш серйозних справ, якими займаються професійні програмісти. Lua має порівняно високу для мов такого класу продуктивність і в неї досить багато бібліотек, хоча і не так багато, як, наприклад, у JavaScript. Нарешті, і це основна причина, чому вона застосовується в OPC-сервері, Lua дуже просто вбудовується, більше того — вона створена для того, щоб її використовували як мову, що вбудовується. Хоча

програми на Lua мають певні обмеження, наявність можливості застосування бібліотек, написаних мовою, Сі цей недолік частково компенсує.

2.4 Вибір серверу для нуль-модемного з’єднання

Для з’єднання між Master OPC і мережею ModBus в межах одного комп’ютера необхідно використосувати емулятор нуль-модемного з’єднання. Емулятор нуль-модему com0com – це відкритий, рівня ядра, драйвер віртуальних послідовних портів для Windows. Драйвер вільно доступний під ліцензією GPL. За допомогою нього можна створити необмежену кількість пар віртуальних COM-портів і використовувати будь-яку пару для з'єднання однієї програми з іншою. Кожна пара містить два COM-порти, які мають за замовчуванням імена виду CNCA<n> та CNCB<n> (перша пара містить порти CNCA0 та CNCB0). Виведення одного порту пари є введення з іншого порту пари і навпаки.

Зазвичай один порт пари використовується програмою Windows, яка потребує наявності COM-порту для взаємодії з пристроєм, а інший порт використовується програмою-емулятором пристрою.

2.5 Вибір мови програмування для реалізації програмного емулятора технологічного процесу разом з промисловою мережею

Як і в бакалаврській роботі, яка була присвячена розробці програмного емулятора мережі MobBus, для виконання цієї задачі доцільно застосувати мову програмування Python. Але в даному випадку обмежимость лише емулятором без RESTful сервісу. Для цього Використаємо дистрибутів Python3 з перевіреними на сумісність бібліотеками – Anaconda [14]. Для роботи зі структурами та колекціями використаємо бібліотеки struct та collections. Для роботи з часовими інтервалами використаємо бібліотеки time та datetime. Для роботи з системними викликами та потоками використаємо бібліотеки os, sys, signal, threading. Для роботи з послідовним портом використаємо бібліотеку (py)serial. Для роботи з протоколом Modbus RTU використаємо бібліотеку modbus_tk. Для отримання діагностичних даних з бібліотек та ведення звітів

використаємо бібліотеку logging. Бібліотека modbus_tk дозволить працювати в режимі майстра й слейва. Бібліотека вимагає pyserial, встановлення якої може визивати конфлікти. Для реалізації текстового інтерфейсу використаємо бібліотеку (windows)-curses. Для реалізації матричних розрахунків використаємо бібліотеку NumPy.

Висновки за розділом

1. Проведено вибір необхідного програмного забезпечення, яке знадобиться для імітаційного моделювання роботи АСУ ТП
2. В якості SCADA-системи виберемо Simple-SCADA, яка має вбудовану мову програмування Free Pascal.
3. В якості OPC-системи виберемо Master OPC, який має вбудовану мову програмування Lua.
4. В якості емулятора нуль-модемного з’єднання виберемо програму com0com
5. Для розробки емулятора технологічного процесу виберемо мову Python та необхідні для цього бібліотеки.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ РОБОТИ АСУТП

3.1 Вибір технологічного процесу для дослідження

Розглянемо технологічний процес двокомпонентної ректифікації ізопропіл-вода. Математична модель процесу наведена в роботі [15]. Переведена в простір станів в дискретним часі з чистим представленням запізнення математична модель має вигляд:

$$A=\begin{bmatrix}
0.99500 & 0.00000 & 0.03125 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000\\
0.00000 & 0.99064 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.03125 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 1.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 1.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 1.00000 & 0.00000 & 0.00000 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 1.00000 & 0.00000 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 1.00000\\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000
\end{bmatrix}$$

$$B=\begin{bmatrix}
0.00000 & 0.00000\\
0.00000 & 0.00000\\
0.00000 & 0.00000\\
0.00000 & 0.00000\\
0.00000 & 0.00000\\
0.00000 & 0.00000\\
1.00000 & 0.00000\\
0.00000 & 0.00000\\
0.00000 & 1.00000
\end{bmatrix}$$

$$C=\begin{bmatrix}
-0.02247 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0.05408 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

D=0

Вибраний крок дискретності $\Delta t = 0.18$ с.

Технологічна схема процесу ректифікації показана на рис. 4.

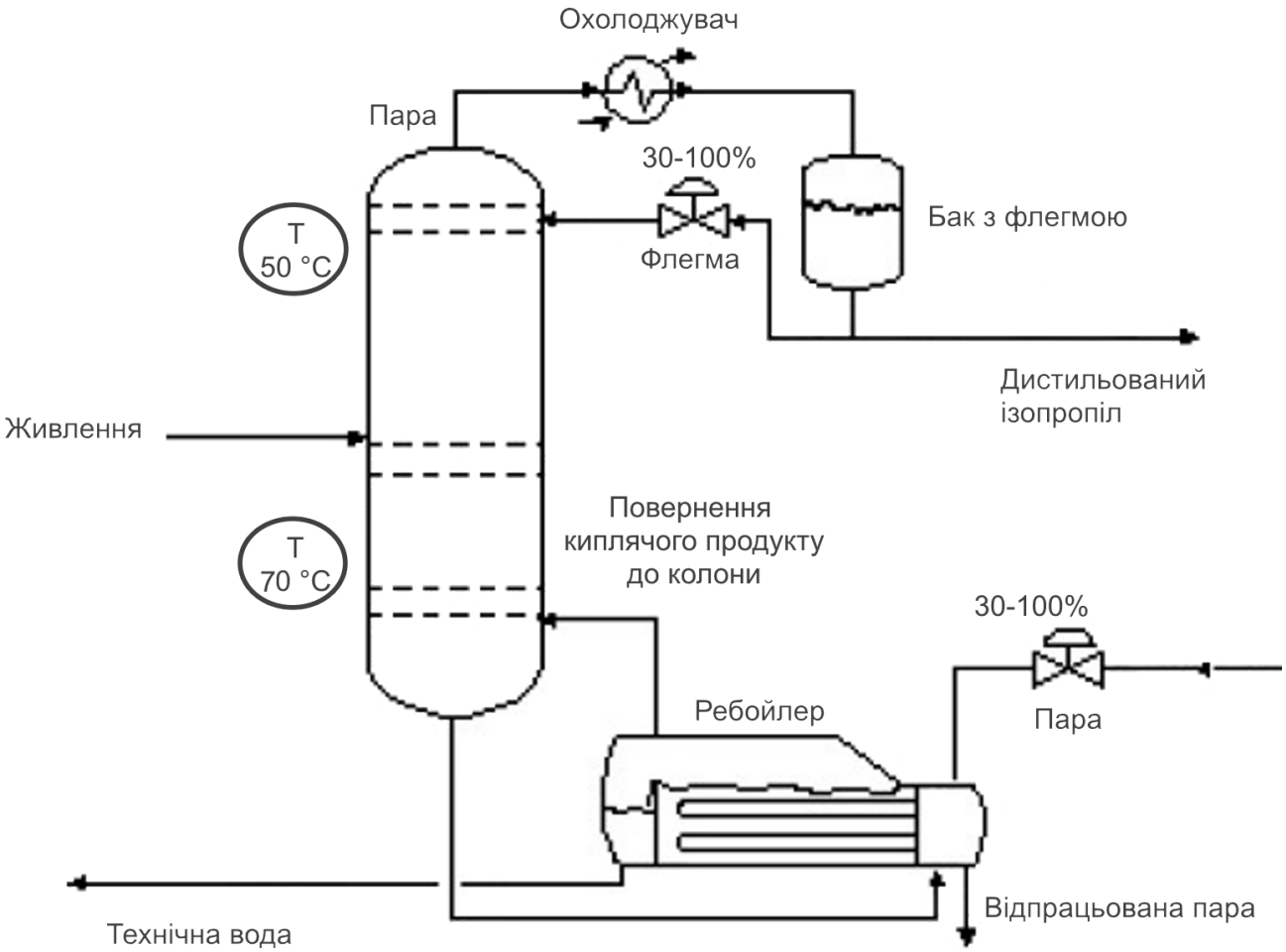


Рисунок 4 – Технологічна схема процесу ректифікації –

Задачею технологічного процесу є як можливо точніше підтримання температури верху (50 °C) і температури низу (70 °C) колони. Для цього необхідно реалізувати систему керування, керуючими впливами в якій може бути витрата пари в ребойлер і витрата флегми в колону. Це робиться за допомогою двигунів які змінють положення регулюючий органів. Їх робочий діапазон знаходиться в межах від 30 до 100%.

3.2 Вибір засобів вимірювання й плат керування двигунами

Критерії вибору засобів вимірювання і плат керування двигунами – висока точність, можливість безпроблемного під’єднання до мережі ModBus, докладна документація, помірна вартість.

Пошук і інтернеті показав, що доцільним вибором для вимірювання температури є давач TXTMINI-M12-485 фірми Novus. Зовнішній вигляд давача температури наведений на рис. 5

Рисунок 5 – Зовнішній вигляд давача температури

Основні технічні характеристики давача наступні:

- 2-х дротовий інтерфейс RS485;
- Живлення від 7 до 40 В.
- 3-х дротове під’єднання до сенсору Pt100;
- Діапазон вимірювання: -200 to 600°C;
- Типова точність вимірювання (при 25 °C): 0.1%;
- Робоча температура: -40 to 85 °C;
- Програма для конфігурації.

Основним робочим регістром давачей є регістр 5, який зберігає значення вимірювань як ціле число. Наприклад, 500 – це 50.1 °C. Більш докладний опис наведено в документації [16], а регістрів – в додатку А.

Для керування виконавчими механізмами використаємо плату TMB1-RTI Modbus Control Board for Electric Actuators фірми A-T Controls, Inc.

Основні можливості плати наступні:

- Можливість роботи під живленням від 115 до 230 В;
- Позиціонування зі зворотнім зв’язком на базі показів потенціометра (давача положення);
- Інтерфейс RS-485 з можливість налаштування формату підключення за допомогою джамперів;

- Можливість керування положенням в ручному режимі безпосередньо за допомогою кнопок і індикаторів на платі.

Основним регістром є регістр 10, за допомогою якого вказується необхідне положення рег. органу, регістр 8 відображає поточне положення, регістр 0 – напрям руху рег. органу (0 – немає, 64 – за часовою стрілкою, 128 – проти). Більш докладний опис наведено в документації [17], а регістрів – в додатку А.

3.3 Розробка програмного симулятора технологічного процесу і польової мережі ModBus

Програмний симулятор реалізовано мовою Python 3. Сама програма має текстовий інтерфейс, який реалізовано з використанням бібліотеки псевдографіки curses.

Зовнішній вигляд програмного симулятора в нормальному робочому режимі показано на рис. 6

Temperature Top : 50.0 Deg C

Temperature Btm : 70.0 Deg C

Reflux VLV : 50.0 %

Steam VLV : 50.0 %

	S1	S2		M1	M2
00	60001	60002	00	0	0
01	30001	30001	01	10	10
02	100	100	02	0	0
03	10	10	03	0	0
05	500	700	05	0	0
07	3	3	07	10	10
08	0	0	08	500	500
09	1	1	10	500	500
10	0	0	11	60	60
11	0	0	12	300	300
12	0	0	13	5	5
13	0	0	14	1000	1000
22	5	5	15	500	500

Column is operating in normal mode

Legend: Primary RW | Primary R | Secondary RW

Рисунок 6 – Зовнішній вигляд текстового інтерфейсу програмного симулятора

Бажаними показниками якості ходу технологічного процесу є підтримання температури зверху близької до 50 °С, температури знизу близької до 70 °С. На технологічний процес діють збурення за живленням колони, тому необхідно використовувати автоматичне керування.

Розроблене програмне забезпечення виконує наступні функції:

1) За вказаним номером COM-порту й швидкістю організує симуляцію в режимі SLAVE ModBus мережі в використанням бібліотеки modbus_tk

2) Створює 4 SLAVE пристрої – два давача і два виконавчих механізми з відповідними налаштуваннями за замовченням

3) В окремому потоці проводить матричне моделювання перехідних процесів в ректифікаційний колоні за допомогою матричного диференціювання. Також моделюється затримка виконавчого механізму (який не може рухатись більш ніж 0.3% за крок дискретності в 0.18 с) і мінливі збурення за живленням, які притаманні технологічному процесу.

4) Текстовий інтерфейс інтерактивно відображує поточне значення давачів і реагує на команди, які приходять до виконавчих елементів від MASTER ModbBus мережі.

5) В випадку надмірного відкриття регулюючих органів симулятор починає відлік до аварії. Після закінчення відлику симулятор автоматично закривається.

Програмний код симулятора наведено в додатку Б.

3.4 Типове налаштування OPC-серверу

Для типового налаштування Matrkion OPC необхідно виконати наступні дії:

1) Створити сервер. Назвемо його Server.

2) В сервері створити вузел, який відповідає певному технологічному агрегату чи ділянці, назвемо вузел DC (від Distillation Column – ректифікаційна колона). Встановимо порт 8 (порт, який зв’язаний за допомогою програми com0com до 4 порту, на якому працює симулятор). Інші налаштування в

типовому випадку можливо залишити за замовчуванням (швидкість, контроль чітності і т.п.)

3) В вузлі DC створимо прилад і назвемо його Temperature Top (температура зверху). Прив’яжемо його до адреси в мережі ModBus 1. Час відповіді (макс) встановимо 100 мс. Період опитування 1000 мс.

4) В приладі Temperature Top створимо тег Measurement (вимірювання). Задамо йому тип даних float і прив’яжемо до 5-го регістру. Встановимо коефіцієнт масштабування A=0.1. Права – тільки для читання.

5) В вузлі DC створимо прилад і назвемо його Temperature Bottom (температура знизу). Прив’яжемо його до адреси в мережі ModBus 2. Час відповіді (макс) встановимо 100 мс. Період опитування 1000 мс.

6) В приладі Temperature Bottom створимо тег Measurement (вимірювання). Задамо йому тип даних float і прив’яжемо до 5-го регістру. Встановимо коефіцієнт масштабування A=0.1. Права – тільки для читання.

7) В вузлі DC створимо прилад і назвемо його Reflux (флегма). Прив’яжемо його до адреси в мережі ModBus 3. Час відповіді (макс) встановимо 100 мс. Період опитування 20 мс.

8) В приладі Reflux створимо тег Moving (Рухається). Прив’яжемо його до 0 регістру. Тип даних встановимо int32. Права – тільки для читання.

9) В приладі Reflux створимо тег MVNow (поточна позиція регулюючого органу). Прив’яжемо його до 8 регістру. Тип даних встановимо float. Встановимо коефіцієнт масштабування A=0.1. Права – тільки для читання.

10) В приладі Reflux створимо тег MV (бажана позиція регулюючого органу). Прив’яжемо його до 10 регістру. Тип даних встановимо float. Встановимо коефіцієнт масштабування A=0.1. Права – для читання і запису.

11) В вузлі DC створимо прилад і назвемо його Steam (пара). Прив’яжемо його до адреси в мережі ModBus 4. Час відповіді (макс) встановимо 100 мс. Період опитування 20 мс.

12) В приладі Steam створимо тег Moving (Рухається). Прив’яжемо його до 0 регістру. Тип даних встановимо int32. Права – тільки для читання.

13) В приладі Steam створимо тег MVNow (поточна позиція регулюючого органу). Прив'яжемо його до 8 регістру. Тип даних встановимо float. Встановимо коефіцієнт масштабування A=0.1. Права – тільки для читання.

14) В приладі Steam створимо тег MV (бажана позиція регулюючого органу). Прив'яжемо його до 10 регістру. Тип даних встановимо float. Встановимо коефіцієнт масштабування A=0.1. Права – для читання і запису.

15) Зберігаємо проєкт.

Якщо все нормально, то після запуску проєкту Matrikon OPC сервер має показати свій робочий екран. Після перезавантаження сервер буде запускатись в фоновому режимі при першому запиті до нього. Приклад робочого екрану показаний на рис.

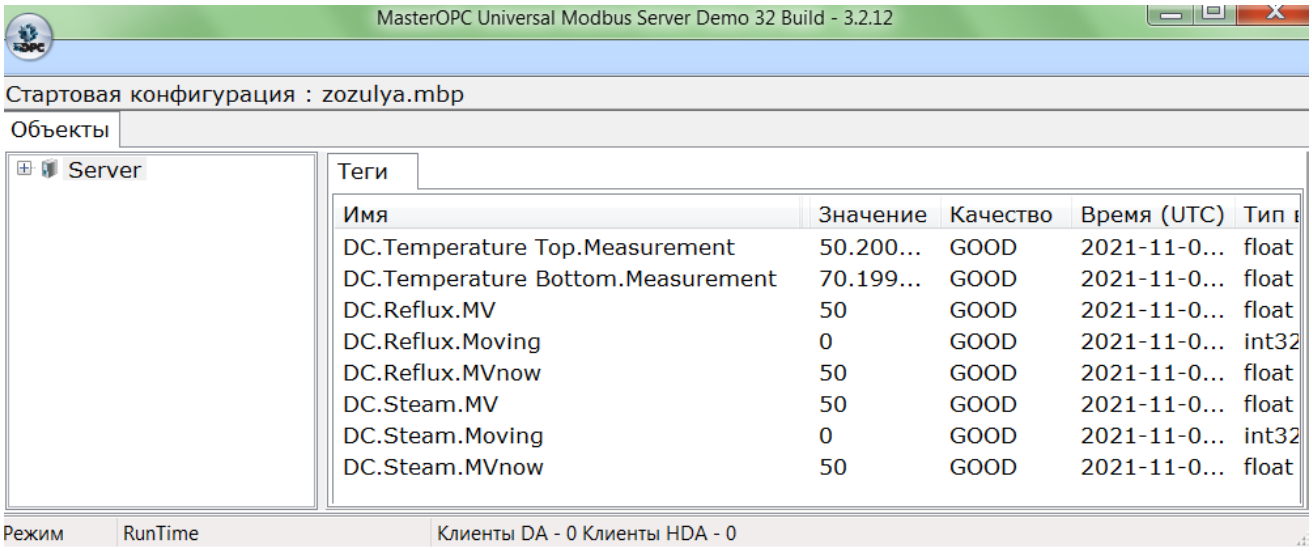


Рисунок 7 – Приклад нормальної роботи OPC-серверу

Тепер OPC-сервер можливо під'єднати до SCADA-системи чи іншого програмного забезпечення АСУТП.

3.5 Підміна OPC-серверу

Перший етап налаштування OPC-серверу на який пройде підміна нормального робиться аналогічно типовому: створюються аналогічні пристрої і теги. Тобто для SCADA-системи не має нічого бути помітним ідентифікатор

серверу такий же, графічний діалог OPC-серверу на перший погляд виглядає таким же. Але внутрішня будова проєкту зовсім інша.

Як було сказано раніше, Matrikon має мову програмування Lua, за допомогою якої і будемо робити підміну. Для реалізації скриптів в мові треба зробити теги дещо іншим чином, не прив'язуючи їх для пристроїв (т.зв. SERVER-теги). Після цього для кожного тегу необхідно прив'язати певний скрипт на мові Lua.

Програмний код скриптів наведено в додатку В.

Відмітимо певні особливості програмування на мові Lua в сервері Matrikon, які роблять роботу доволі незручною:

1) відсутні глобальні функції і змінні (не в самій мові, а в її конкретній реалізації)
2) відсутні глобальні функції, що вимагає повторювати код
3) програмні бібліотеки ModBus реалізовані з помилкою, тому кодування цілого числа прийшлось писати вручну й записувати його як строку стандартною функцією з бібліотеки Modbus OPC-серверу.

3.6 Розробка людино-машинного інтерфейсу в SCADA-системі

Розробка людино-машинного інтерфейсу (ЛМІ) починається зі створення проєкту. Проєкт – це набір внутрішніх і зовнішніх (під'єднаних до OPC-серверу) тегів, екрани і графічні елементи, скрипти і засоби доступу і зберігання в СКБД. Графічна частина проєкту має один основний екран, з яким має працювати оператор. Далі ще розробимо екран для відлагодження програмного забезпечення системи захисту і системи регулювання, але в цьому підрозділі це описувати не будемо.

Спочатку потрібно розробити графічні елементи і завантажити їх в SCADA-систему за допомогою програми pictures. В даному випадку необхідно розробити два типи елементів: мнемосхему і зображення для сигналізації. Розроблена мнемосхема показана на рис. 8, індикатори – на рис. 9. Для того,

щоб не розміщати на мнемосхемі два індикатори в програмі pictures індикатори необхідно розбити на кадри.

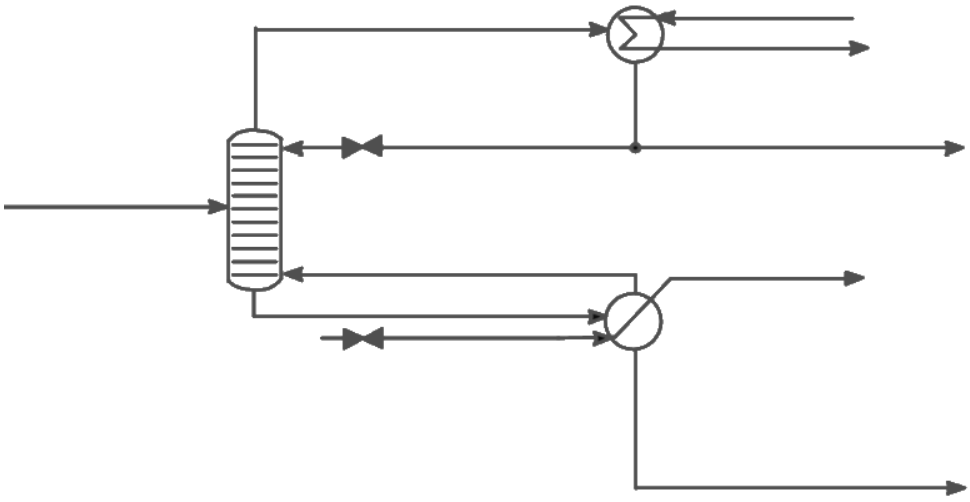


Рисунок 8 – Розроблена мнемосхема технологічного процесу

Рисунок 9 – Індикатори ненормального проведення технологічного процесу, які з’являються на мнемосхемі поряд зі змінною, значення якої дійшло передаварійного чи аварійного рівня

Зовнішній вигляд повністю сформованого екрану ЛМІ показано на рис. 10

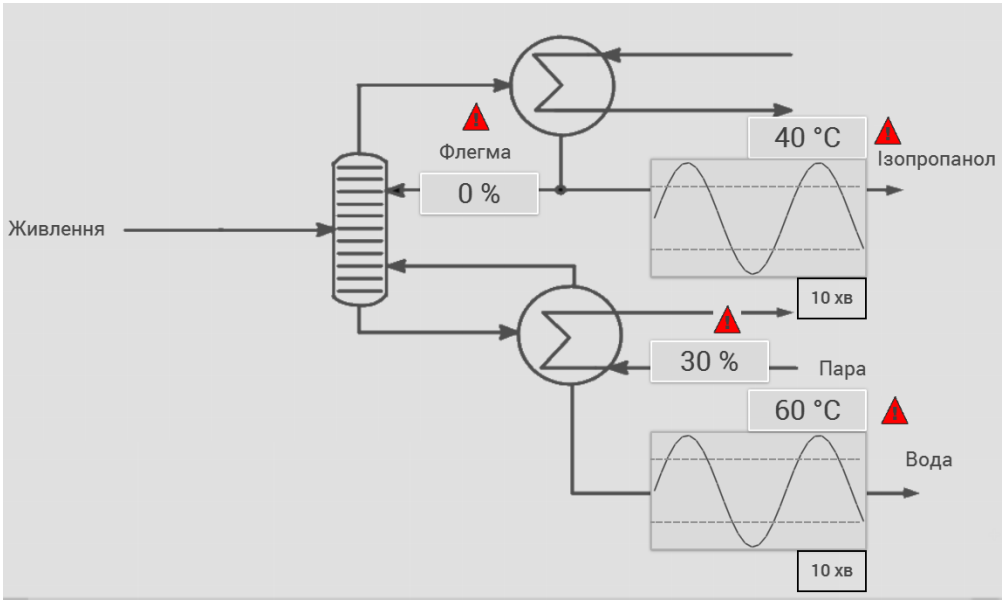


Рисунок 10 – Розроблений основний екран ЛМІ

Екран, який представлено на рис. 10 складається з елементів, зміст яких описаний в таблиці 1.

Таблиця 1 – Елементи основного вікна ЛМІ

Ідентифікатор	Тип	Прив'язка до змінної	Властивості/Пояснення
imgMain	TM_IMAGE	–	Головна мнемосхема
ind2_reflux			Індикатори порушення режими витрати флегми, пари, температури зверху і знизу За замовченням Visible=False
ind2_steam			
ind2_ctop			
ind2_cbtn			
txtFeed	TM_TEXT		Підпис «Живлення»
txtRelux			Підпис «Флегма»
txtIsopropanol			Підпис «Ізопропанол»
txtWater			Підпис «Вода»
txtSteam			Підпис «Пара»
fldU1	TM_FIELD	var_u1	Поле для виводу значення витрати флегми
fldU2		var_u2	Поле для виводу значення витрати пари
fldY1		var_y1	Поле для виводу температури зверху
fldY2		var_y2	Поле для виводу температури знизу
trendY1	TM_TIME TRENDVIEWER	var_y1_normed	Вивід нормованих до шкали -5..5 температур на безшкальних трендах з вказаними межами
trendY2		var_y2_normed	

Перелік тегів SCADA наведений в таблиці додатку Г.

Програмний код для індикації і сигналізації наведений в додатку Д.

3.7 Розробка системи автоматичного керування і ведення журналу даних в SCADA-системі

Для нашого дослідження висока якість керування не є критичною, тому для керування використаємо типові ПІ-регулятори. Для визначення налаштувань регуляторів скористаємось пакетом Matlab Simulink. Схема моделювання представлена на рис. 11.

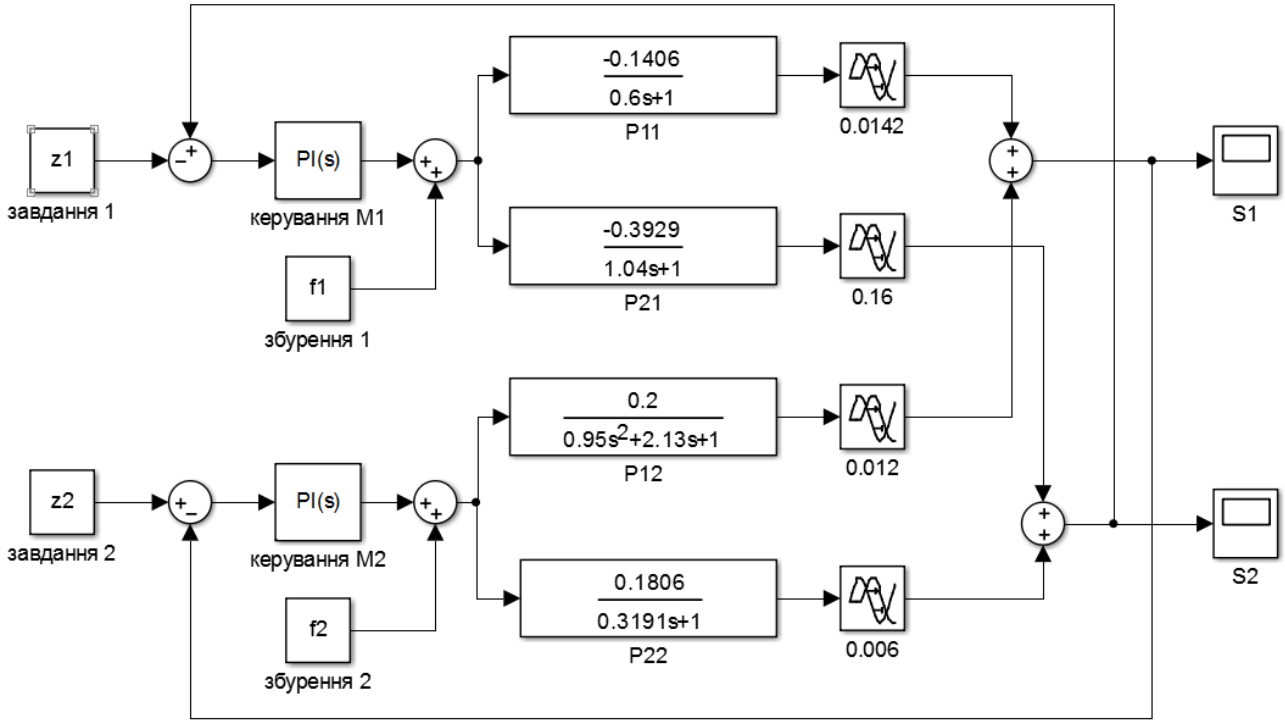


Рисунок 11 – Модель системи керування ректифікаційною колоною (час в хвилинах)

Керовані змінні – температури зверху та знизу колони. Керуючі впливи – витрати пари і флегми (готового продукту, який повертається до колони для покращення розділення). S1 – це давач температури зверху, S2 – давач температури знизу, M1 – виконавчий механізм витрати флегми, M2 – виконавчий механізм витрати пари. Бажаними показниками якості ходу технологічного процесу є підтримання температури зверху близької до 50 ^{o}C, температури знизу близької до 70 ^{o}C. На технологічний процес діють збурення за живленням колони, тому використовується автоматичне керування.

Закон керування в ПІ-регуляторі в цифровій формі реалізується за формою:

$e = r_i - y_i;$

$u = u_i + Kp \cdot e_i + Kp \cdot Ki \cdot e_{i-1};$

$u = sat(-20, 50),$

де *i*-номер кроку, *e*-помилка керування, *y*-значення керованої змінної, *r* - значення завдання на керовану змінну, *Kp*-налаштування пропорційної складової регулятора, *Ki*-налаштування інтегральної складової регулятора, *sat* – обмеження.

Для відлагодження системи автоматичного керування в SCADA-системі розроблено спеціальний екран, показаний на рис. 12

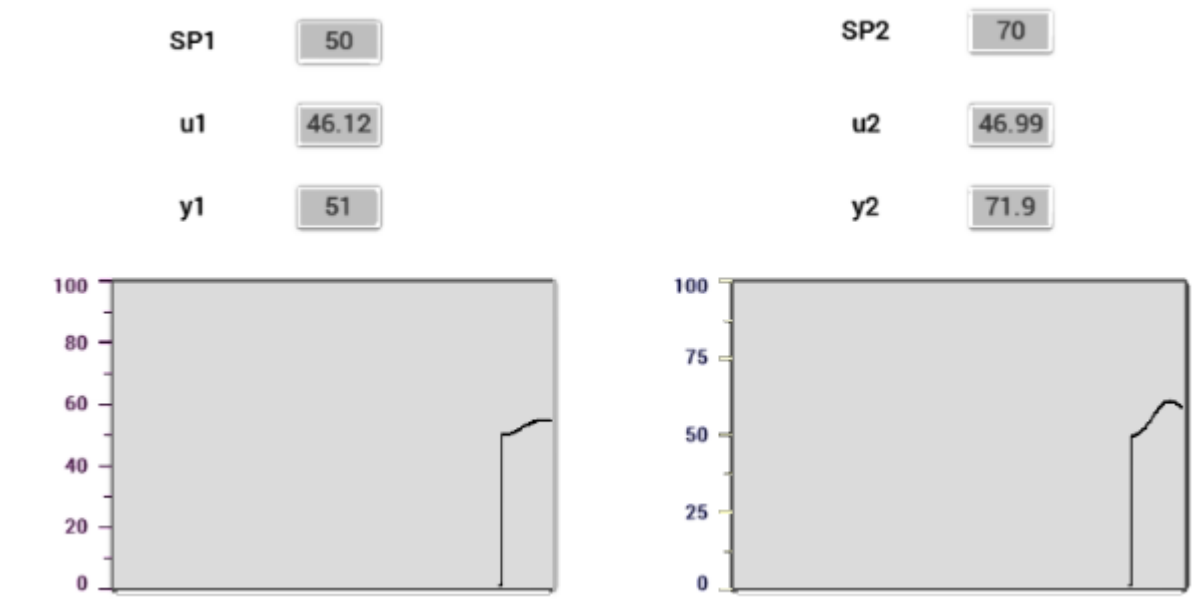


Рисунок 12 – Екран відлагодження системи автоматичного керування

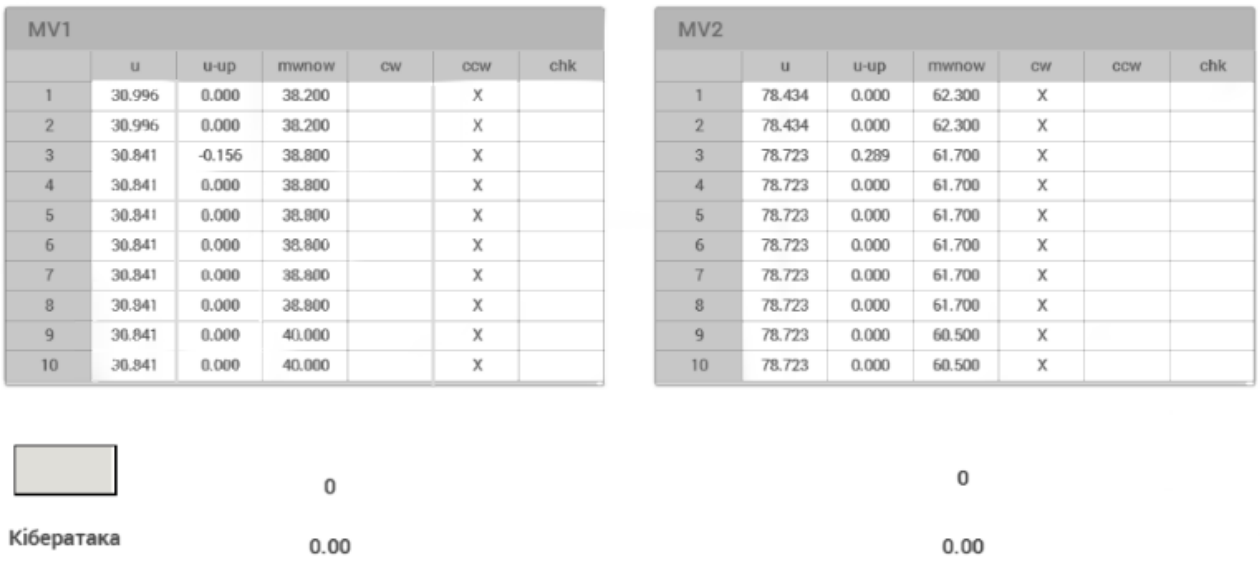
Також для ведення журналу даних реалізовані глобальні масиви (модуль Glb), які зберігають десять останніх значень розрахованих регуляторами керувань, десять різниць між поточним і попередніми значеннями розрахованих регуляторам керувань, десять поточних значень положень регулюючих органів, і десять останніх напрямків руху регулюють.

Програмний код, який реалізує керування і збереження історії керуючих впливів, приведений в додатку Е.

3.8 Розробка системи кіберзахисту в SCADA-системі

В програмному забезпеченні SCADA-системи має бути реалізовано алгоритм виявлення зламу. Сутність цього алгоритму полягає в тому, що для кожного регулятору розраховується різниця між поточним й попереднім (секунду назад) значенням керуючого впливу. Якщо $u_i - u_{i-1} > 0$ і значення регістру 0 дорівнює 64, або $u_i - u_{i-1} < 0$ і значення регістру 0 дорівнює 128, або $u_{i-1} = 0$ і значення регістру 0 дорівнює 0, то процес керування йде вірно. При порушенні будь-якого з цих умов підраховується кількість таких порушень і якщо вона перевищує порогове значення (прийнято 10 підряд), то спрацьовує сигналізація в SCADA-системі (червоний прямокутник) й оператору передається повідомлення про хакерську атаку.

Для відлагодження системи кіберзахисту в SCADA-системі розроблено спеціальний екран, показаний на рис. 13.



MV1	u	u-up	mwnow	cw	ccw	chk
1	30.996	0.000	38.200		X	
2	30.996	0.000	38.200		X	
3	30.841	-0.156	38.800		X	
4	30.841	0.000	38.800		X	
5	30.841	0.000	38.800		X	
6	30.841	0.000	38.800		X	
7	30.841	0.000	38.800		X	
8	30.841	0.000	38.800		X	
9	30.841	0.000	40.000		X	
10	30.841	0.000	40.000		X	

MV2	u	u-up	mwnow	cw	ccw	chk
1	78.434	0.000	62.300	X		
2	78.434	0.000	62.300	X		
3	78.723	0.289	61.700	X		
4	78.723	0.000	61.700	X		
5	78.723	0.000	61.700	X		
6	78.723	0.000	61.700	X		
7	78.723	0.000	61.700	X		
8	78.723	0.000	61.700	X		
9	78.723	0.000	60.500	X		
10	78.723	0.000	60.500	X		

Рисунок 13 – Екран відлагодження системи кіберзахисту

Екран розділений на дві частини – для першого і другого керуючого пристрою. В першому стовпці розміщено значення керуючого впливу, розраховане регулятором, в другому – різниця між поточним і попереднім значенням керуючим впливом регулятора, в третьому – поточне значення давача положення регулюючого органу, в четвертому і п’ятому – рух регулюючого органу в напрямку часової стрілки чи проти нього. Знизу – перший рядок – це лічильник відсутності коректного руху регулюючого органу, другий рядок – різниця між поточним і попереднім керуючим впливу регулятора. Якщо лічильник (який розраховується кожну секунду) перевищує значення 10, то спрацьовує сигналізація.

Програмний код системи кіберзахисту приведено в додатку Ж.

3.9 Перевірка роботи системи кіберзахисту в SCADA-системі

Розглянемо як мету хакерської атаки зупинку технологічного процесу. Для цього хакеру необхідно відключити автоматичне керування і вивести витрату флегми та пари на 100% відкриття.

Запустимо комп’ютерне програмне забезпечення системи керування – SCADA-систему (рис. 14). Ця система реалізує процес керування і людино-машинний інтерфейс. Зв'язок з промисловою мережею в SCADA-системі реалізується за допомогою OPC-серверу. Вважаймо, що доступ до SCADA-системи хакер не має.

Оскільки на процес діють збурення (які моделюються в симуляторі) значення керованої змінної не буде постійним, тому після запуску SCADA-системи почнуться перехідні процеси, доки система не буде стабілізована, тобто значення температури зверху не буде 50 °С, а температури знизу – 50 °С.

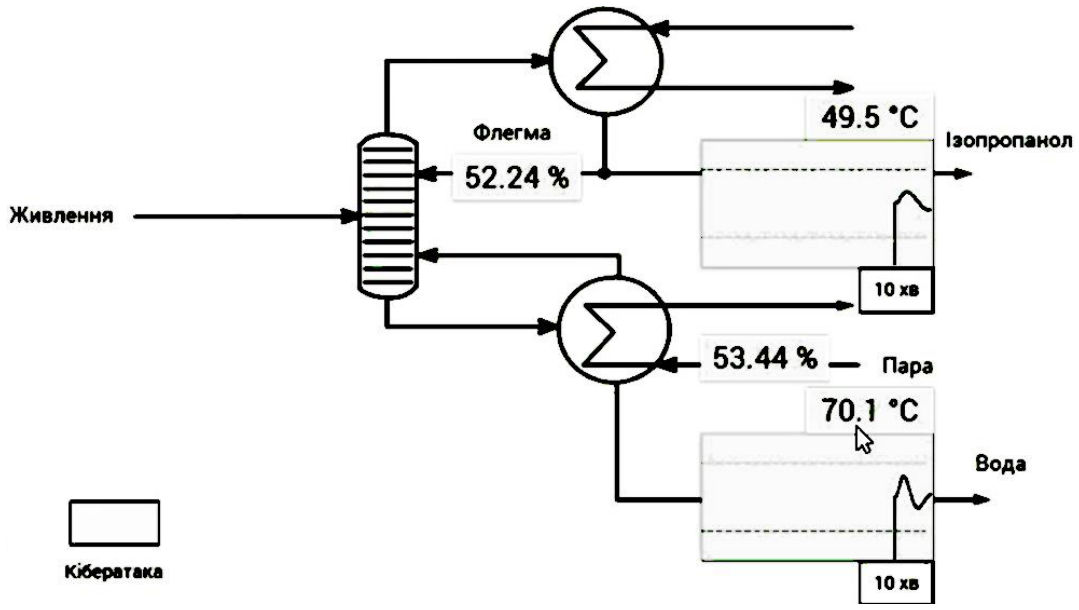


Рисунок 14 – Працююча форма оператора-технолога в SCADA-системі

Дочекаємось часткової стабілізації параметрів технологічного процесу. Бачимо за трендами на рис. 14, що регулятори реалізують автоматичне керування.

Метою хакера є створити видимість того, що автоматичне керування продовжується і змінні більш-менш в регламентних межах. В той же час значення не мають бути постійними, бо для оператора технологічної установки це буде дуже помітним.

Хакер не має моделі технологічного процесу і не знає налаштувань регуляторів, тому для нього єдиним виходом щоб не видати себе є програмна імітація достатньо реалістичного продовження ходу технологічного процесу після зламу таким чином, щоб регулятори за заданий час не вийшли за регламентні межі. Це він реалізує шляхом заміни OPC-серверу на такий же, але перепрограмований під фальсифікацію показань давачей і положень виконавчих механізмів.

Проведемо підміну OPC-сервера, запустивши скрипт hack з командної консолі Windows (рис. 15).

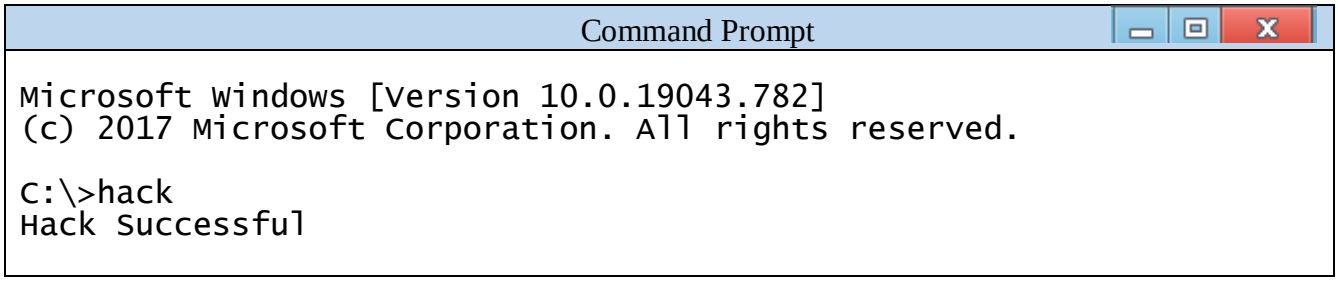


Рисунок 15 – Введення команди для зламу

Після цього станеться неконтрольоване збільшення положення виконавчих механізмів (бачимо що завдання на реальне положення виконавчих механізмів встановлені на 100% відсотків у вікні симулятора на рис. 16). Але на формі SCADA-системи цього помітно не буде (рис. 14).

	S1	S2		M1	M2
08	0	0	08	710	710
09	1	1	10	1000	1000

Рисунок 16 – Значення регістрів в симуляторі після зламу

В програмному забезпеченні SCADA-системи працює алгоритм виявлення зламу, сутність якого описана в попередньому підрозділі.

Бачимо, що OPC-сервер передав у промислову мережу значення керуючих впливів, яке дорівнює 100% (рис. 14). Відлік часу до порушення роботи колони почався. Програмне забезпечення, реалізоване в SCADA має виявити нестандартну поведінку. Критерієм нестандартної поведінки є відсутність руху регулюючого органу при підтверджені зміни положення.

На формі SCADA-системи бачимо, що процес для оператора установки висвітлюється як нормальний (рис. 17). Але розроблена система діагностики виявила злам і пропонує оператору технологічної установки зупинити OPC-сервер. Це він повинен зробити з допомогою спеціальної кнопки. Після її натискання оператором OPC-сервер буде зупинено разом зі SCADA-системою.

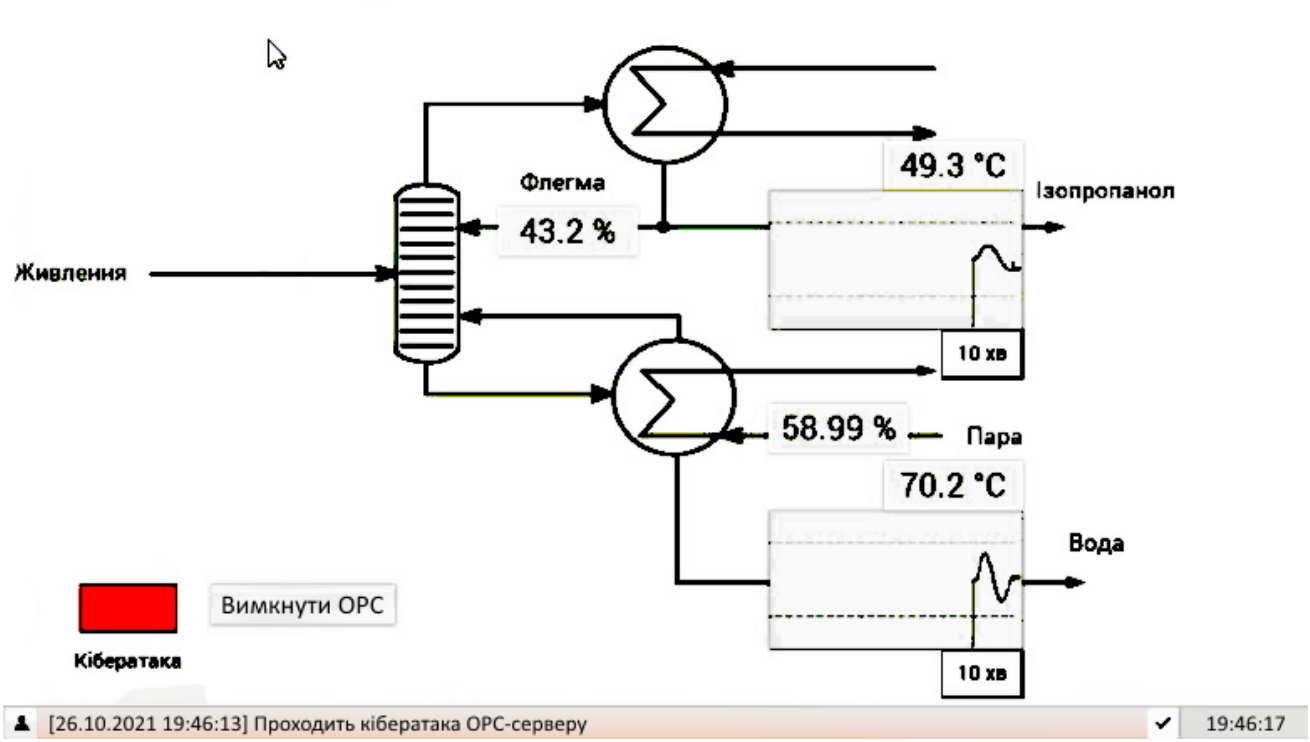


Рисунок 17 – Екран SCADA-системи після діагностування кібератаки

Після відімкнення SCADA-системи адміністратор мережі має повернути положення регулюючих органів за допомогою команд Modbus чи оператор технологічної установки має повернути положення регулюючих органів вручну за допомогою ручного дублера, який встановлений на кожному регулюючому органі. Далі системний адміністратор мережі повинен зрозуміти знаходження джерело зламу, усунити його та відновити роботу оригінального ОРС-серверу. Після цього SCADA-систему треба запустить знову.

Висновки по розділу

1) В якості технологічного процесу розглянуто процес розділення суміші ізопропанолу з водою.

2) Проведено вибір необхідних засобів автоматизації, які необхідні для мінімального керування технологічним процесом. Всі обрані засоби автоматизації мають інтерфейс доступу до ModBus мережі.

3) Використовуючи математичну модель динаміки ректифікаційної колони і технічну документацію засобів автоматизації проведено розробку симулятора технологічного процесу мовою Python 3.

4) Проведено налаштування зв’язку ModBus OPC серверу через нуль-модемне з’єднання з програмним симулятором.

5) За допомогою типового пакету SCADA реалізовано людино-машинний інтерфейс, система автоматичного керування, система виявлення зламу.

6) Промодельовано злам шляхом підміни OPC серверу з продовженням симуляції нормального ходу технологічного процесу. Результати моделювання показують, що система кіберзахисту спрацювала и швидко виявила підміну.

4 ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.

Охорона здоров’я трудящих ,профілактика професійних захворювань, забезпечення умов праці –все це має важливе значення на будь-якому виробництві, зокрема, на робочому місці інженера-спеціаліста з інформаційної безпеки. На жаль, науково-технічний прогрес не усуває ймовірності несприятливого впливу на людину та навколишнє природне середовище в процесі експлуатації, обслуговування, ремонту, налагодження, випробування обладнання. Діяльність людини у сучасних умовах супроводжується підвищенням психофізичної напруженості, збільшенням інформаційного та нервово-емоційного навантаження , появою нових фізичних видів впливу на працюючих. Має місце також обмеження рухливості людей, нерівномірність м'язового та психологічного навантаження , що сприяє розвитку втоми та створення передумов для травматизму захворювання та помилок.

Головна мета охорони праці - підготовка та реалізація заходів, які сприяють забезпеченню здорових умов праці, утриманню в належному стані робочого місця, страхування працівників.

Робоче місце інженера-спеціаліста з інформаційної безпеки має наступне обладнання:

- стіл;
- крісло;
- персональний комп'ютер у стандартній комплектації -системний блок, клавіатура, монітор, пристрої виведення інформації та ін.;
- принтер.

Оскільки основне робоче положення інженера-фахівця з інформаційної безпеки -сидячи, отже, робоче місце має бути організовано відповідно до існуючих певних стандартів. Те, що потрібно для виконання робіт частіше, має бути у зоні легкої досяжності робочого простору.

У процесі роботи інженер-програміст може зазнавати впливу шкідливих та небезпечних факторів:

Небезпека ураження електричним струмом. Викликана вона тим, що усе обладнання живиться від трифазної мережі змінного струму (380/220, 50 Гц). У процесі експлуатації людина може торкнутися частин, що є під напругою. Дія електричного струму на організм людини може спричинити травми різного ступеня тяжкості та навіть смертельний результат.

Підвищений рівень статичної електрики. Електризація - це комплекс фізичних і хімічних процесів, що призводять до поділу в просторі протилежних зарядів знаків або до накопичення зарядів одного знака. Статична електрика шкідливо впливає на організм людини, причому не тільки при безпосередньому контакті з зарядом, але і за рахунок дії електричного поля, що виникає навколо заряджених поверхонь.

Недостатня освітленість робочого місця. Зазвичай вона пов'язана з неправильним вибором та розміщенням освітлювальних приладів у виробничому приміщенні. Умови діяльності інженера-спеціаліста з інформаційної безпеки пов'язані з перевантаженням органів зору. Неякісне освітлення включає знижену контрастність, пряму та відбиту блискітку та підвищену пульсацію світлового потоку. Підвищена яскравість світла сприяє швидкій стомлюваності очей, що призводить як до втрати працездатності, так і до збільшення нервово-психічних навантажень.

Підвищений рівень шуму на робочому місці обумовлений роботою друкувальних пристроїв, накопичувачів, кондиціонерів та систем вентиляції знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, знижує здатність швидко та точно виконувати координовані рухи, зменшує на 5-12% продуктивність праці. Медичні обстеження показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення слуху та появи приглухуватості.

Несприятливі мікрокліматичні умови. Мікроклімат на робочому місці визначається температурою повітря, відносною вологістю, швидкістю руху

повітря, барометричним тиском та інтенсивністю теплового випромінювання від нагрітих поверхонь. Несприятливі мікрокліматичні умови призводять до погіршення самопочуття працівника, ослаблення уваги, швидкої стомлюваності, і при тривалому впливі можуть викликати різні захворювання. Оптимальні комфортні умови - це температура повітря 20-25 градусів, при відносної вологості повітря 40-60 %. Регулярне кондиціювання та хороша вентиляція необхідні для створення нормальних умов для роботи інженера з інформаційної безпеки.

Вплив електромагнітного та м'якого рентгенівського випромінювань. Цей шкідливий фактор може призвести до поступового погіршення зору та інших професійних захворювань інженера-програміста. Джерелом м'якого рентгенівського випромінювання у обчислювальному центрі є відео монітори, тому необхідно дотримуватися відстані не менш 50 сантиметрів до екрану монітора.

Психофізичні навантаження. На жаль, діяльність інженера-спеціаліста з інформаційної безпеки тісно пов'язана з ризиком психологічного навантаження. Розумові та емоційні навантаження обумовлені специфікою праці інженера-програміста: порівняно з фізичною роботою зростає споживання мозком кисню у 15-20 разів. Значна нервово-емоційна напруга може призвести до захворювань серцево-судинної системи. У зв'язку з цим велике значення має планування процесу праці, з метою недопущення перенапруження органів чуття, що призводить до стресів.

Підвищений рівень інфразвукових коливань. Виникає при неправильній експлуатації громіздкої техніки, наприклад, кондиціонера, або при несправному її стані. Тому при експлуатації техніки, зокрема кондиціонерів ,слід дотримуватися усіх вимог щодо їх встановлення та використання.

Біологічно- небезпечні та шкідливі фактори включають такі біологічні об'єкти: патогенні мікроорганізми, бактерії, віруси, спірохети, гриби, тощо. Деякі з перерахованих мікроорганізмів та вірусів здатні завдавати шкоди людині, бути причиною хвороб та втрати працездатності. Тому у місцях зосередження людей

особливо важливою є профілактика захворювань. Також під час пандемії коронавірусної інфекції важливе місце займає вакцинація працівників підприємства.

4.2 Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів

У зв'язку з використанням все більших енергопотужностей людина змушена концентрувати енергію на невеликих ділянках у межах міст та неселених пунктів. Внаслідок цього підвищується ризик виникнення техногенних катастроф, аварій на виробництві. До речі, дуже великі екологічно-небезпечні наслідки має електронна промисловість. Наприклад, виробництво комп'ютерів потребує значної кількості енергії та води, а скорочення терміну дії служби комп'ютерів за рахунок швидкого морального старіння потребує ще й утилізації відходів.

Загалом, техногенні небезпеки за своєю чисельністю виходять на перше місце серед інших. Техногенні небезпеки можуть бути- механічні, хімічні, енергетичні. Механічні небезпеки створюють рухомі об'єкти, колючі та ріжучі предмети, слизькі поверхні та ін. Хімічні небезпеки- забруднення навколишнього середовища небезпечними хімічними речовинами. Енергетичні небезпеки- шуми, вібрації, випромінювання, вибухи, пожежі.

Оскільки інженер-спеціаліст з інформаційної безпеки має справу із електричними приладами, установками та обладнанням, усе це підвищує ризик небезпеки впливу електричного струму. Електричний струм, проходячи через тіло людини, викликає ураження організму, а також, порушує діяльність серцево-судинної системи та системи дихання.

У робочих приміщеннях де працює спеціаліст з інформаційної безпеки є велика кількість легкозаймистих матеріалів. З пожежної безпеки такі приміщення відносяться до категорії "В" . Пожежна безпека забезпечується, по перше - системою запобігання пожежі, а по друге - системою пожежного захисту. Пожежі на підприємстві можуть виникати також внаслідок ушкодження

електропроводки та машин під напругою. Більш ніж 65% пожеж у промисловості виникають унаслідок помилок людей або їх некомпетентності. Щоб уникнути ураження електричним струмом, виникнення пожежі та пошкодження ПК необхідно дотримуватися наступних заходів електропожежної безпеки:

- забороняється експлуатація ПК із несправним шнуром живлення;
- забороняється експлуатація ПК у приміщенні з високою вологістю або сильно забрудненим повітрям;
- не допускається попадання всередину ПК та периферії сторонніх предметів, рідин та сипких речовин;
- не допускаються перегини, передавлювання і натягу кабелів живлення;
- не допускається встановлювати ПК поблизу джерел тепла;
- не допускається закривання вентиляційних отворів ПК та периферії.

- електроживлення робочого місця має бути підключене через рубильник, встановлений у місці, зручному для швидкого відключення живлення робочого місця, а також мають бути вжиті заходи для знеструмлення робочого місця в аварійних режимах;

- приміщення з ПК мають бути оснащені вогнегасниками. Ручні вуглекислотні вогнегасники встановлюються у приміщеннях з обчислювальним обладнанням із розрахунку один вогнегасник на 40-50м2 площі, але не менше двох у приміщенні.

  Комплекс заходів пожежогасіння включає в себе декілька способів: спосіб охолодження, спосіб розбавлення, спосіб ізоляції, спосіб гальмування реакції, спосіб механічного зриву полум’я.

  Своєчасний виклик пожежних підрозділів та оповіщення про пожежу дозволяє локалізувати наслідки пожежі та провести евакуацію людей .В обов’язковому порядку необхідно оснащувати підприємства системами автоматичної пожарної сигналізації та оповіщення.

Крім вищезгаданих факторів ризику на виробництві, на жаль, існує ще один небезпечний чинник ,який впливає на працівників, це збудник COVID-19 – коронавірус 5AK5-CoV-2. Тому, кожен працівник має дотримуватися певних інструкцій щодо запобігання росповсюдження цього небезпечного вірусу. Температурний скрінінг, дезинфекція поверхонь, носіння захисних масок, провітрювання приміщення та, звичайно, обов'язкова вакцінація повинні стати невід'ємною частиною боротьби з короновірусною інфекцією на кожному виробництві.

4.3 Дослідження психомоторних показників і оцінка сили нервової системи шляхом експрес-діагностики за методикою теппінг-тесту.

Дослідження направлено на виявлення психомоторних показників і оцінки сили нервової системи інженера-спеціаліста з інформаційної безпеки

Теппінг-тест визначає витривалість нервової системи, тому обов'язковою умовою виконання тесту є робота в максимальному темпі. Якщо ця умова не виконується, діагностика буде хибною. Обов'язкова умова діагностування сили нервової системи за теппінг-тестом – максимальна мобілізованість обстежуваного. Вкрай важливо починати виконання необхідних дій відразу в максимальному темпі, інакше тест не дає правильних результатів. Теппінг-тест використовується зазвичай у комплексі з іншими тестами, що вимірюють різноманітні характеристики особистості. Особливо корисний такий тест під час профорієнтації та проведення психологічного консультування з корекції і вдосконалення персонального стилю діяльності.

Номер поля	Проміжок часу, с	Кількість точок правою рукою N	Темп руху руки Т, точок/с	Кількість точок лівою рукою N	Темп руху руки Т, точок/с

1	0 – 5	36	7.2	23	4.6
2	6 – 10	27	5.4	19	3.8
3	11 – 15	27	5.4	18	3.6
3	16 – 20	27	5.4	20	4
4	21 – 25	27	5.4	25	5
5	26 – 30	28	5.6	20	4

Таблиця 4.1. Обробка результатів теппінг-тесту

Коефіцієнт сили нервової системи для правої руки

$$K_{CHC}=\frac{(27-36)+(27-36)+(27-36)+(27-36)+(28-36)}{36} = -122.2\%$$

Коефіцієнт сили нервової системи для правої руки

$$K_{CHC}=\frac{(19-23)+(18-23)+(20-23)+(25-23)+(20-23)}{23} = -56\%$$

Коефіцієнт функціональної асиметрії працездатності лівої і правої рук

$$K_{\text{ФА}}=\frac{172-125}{172+125} = 0.23$$

Результат показав, що в обох випадках, як для правої руки так і для лівої, можна зробити висновок, що згідно таблиці яку було приведено в методичці, а саме - Таблиця оцінки нервової системи – розряд і характеристика моєї нервової системи підпадають за значеннями до п’ятого розряду - Дуже висока вираженість сили або слабкості нервової системи.

Щодо коефіцієнту функціональної асиметрії працездатності лівої і правої рук, можна зробити наступні висновки – оскільки отриманий коефіцієнт має знак «+», це свідчить про зміщення балансу в бік збудження.

Якщо визначати тип нервової системи за графіками, то

Для правої руки це -

в) Спадний тип кривої: взятий досліджуваним максимальний темп знижується вже з другого 5-секундного відрізка й залишається на

зниженому рівні протягом усієї роботи. Цей тип кривої свідчить про слабкість нервової системи досліджуваного.

Для лівої руки це -

а) Опуклий тип кривої: темп наростає до максимального в перші 10 – 15 секунд роботи; на проміжку 20 – 30 секунд він може знизитися нижче вихідного рівня, тобто того рівня, який спостерігався в перші 5 секунд роботи. Цей тип кривої свідчить про наявність у досліджуваного сильної нервової системи.

Час виконання тесту	Середній темп руху правої руки Тср п, точок/с	Середній темп руху лівої руки Тср л, точок/с
8.00	5.7	2.96
10.00	5.4	3.6
12.00	5.7	4.2
14.00	5.36	4.12
16.00	6.8	5.04
18.00	6.3	2.03
20.00	5	3.73

Таблиця 4.2. Обробка результатів теппінг-тесті протягом дня

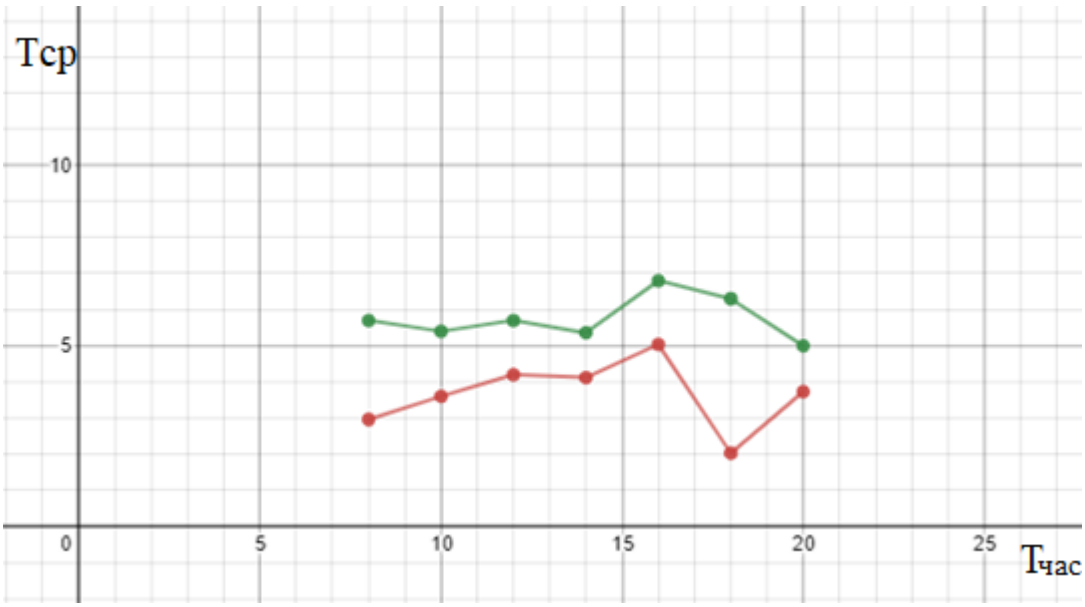


Рисунок 4.1. Графік залежності темпу від часу дня

Зелений графік – права рука.

Червоний графік – ліва рука.

Тести були проведені в повному обсязі, тому можна роботи висновки.

За результатами моїх тестів, які проводилися протягом дня кожні дві години можна сказати наступне - динаміка працездатності ближче до вечора вище ніж з ранку, проте в останньому тесті результат був значно нижче інших тестів проведених раніше, що говорить про те що в пізній час працездатність індивідуума йде на спад. Якщо говорити про планування робочого дня з урахуванням проведених досліджень - робочі години які будуть оптимальні в моєму випадку це - з 10-11 годин ранку до 19-20 години вечора.

ВИСНОВКИ

За результатами аналізу літератури встановлено, що захист промислових АСУТП є актуальною і не дуже розвинутою науковою проблемою. В той же час аналіз цілей хакерів та реальних фактів зламу показує, що промисловий сектор є один з найбільшою ціллю кібератак в розвинутих країнах. Причина наявності значних проблем значною мірою полягає в тому, що при розробці програмного забезпечення АСУТП як правило не залучались спеціалісти з кібербезпеки, тому таке програмне забезпечення потенційно має значну кількість уразливостей, значну кількість яких знаходять кожен рік за допомогою нескладних інструментів (фаззінг та ін.), якими володіє будь-який хакер.

Як правило при розгляданні задачі кіберзахисту промислових АСУТП розглядають питання захисту контролерів чи SCADA. В цій роботи запропоновано розглянути стратегію кібератаки, яка може використосувати лише OPC-сервер, який теж є доволі вразливим елементом. При якісному проведенні такої атаки оператор нічого не помітить до того часу, доки технологічний об’єкт не буде виведений в аварійний режим роботи.

Для підтвердження можливості такої атаки і розробки методу її діагностики були вибрані необхідні програмні інструменти, в тому числі програмне забезпечення, яке є типовим для виробничих АСУТП.

Серед типового програмного забезпечення в якості SCADA-системи обрана Simple-SCADA, яка має вбудовану мову програмування Free Pascal. В якості OPC-системи обнрано Master OPC, який має вбудовану мову програмування Lua.. В якості емулятора нуль-модемного з’єднання обрано програму com0com.

Була проведена розробка емулятора технологічного розділення ізопропіла з водою з використанням мови програмування Python. Емулятор був під’єднаний до OPC-серверу та налаштованої SCADA-системи.

Проведене моделювання в реальному часі показало можливість розглянутої атаки та успішне спрацювання системи діагностики кібератаки.

ПЕРЕЛІК ПОСИЛАНЬ

1 https://www.infosecurity-magazine.com/news/industrial-orgs-penetrated-hackers/

2 https://www.washingtonpost.com/business/2021/05/08/cyber-attack-colonial-pipeline/

3 https://www.infosecurity-magazine.com/news/cybercrime-costs-orgs-per-minute/

4 https://standoff365.com/

5 https://www.fireeye.com/content/dam/fireeye-www/products/pdfs/wp-top-20-cyberattacks.pdf

6 https://orignix.com/what-are-the-provisions-of-isa-iec-62443/

7 https://isup.ru/upload/pdf-zhurnala/2018%20i%20dalee/2018/4_76_2018/013_015_AdAstrA.pdf

8 https://cyberx-labs.com/wp-content/uploads/2020/09/CYBX_2020_Risk-Report.pdf

9 https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/07/07190426/KL_REPORT_ICS_Statistic_vulnerabilities.pdf

10 https://ics-cert.kaspersky.ru/news/2021/03/04/more-critical-vulnerabilities-identified-in-opc-protocol-implementations/

11 https://ics-cert.kaspersky.ru/reports/2018/05/10/opc-ua-security-analysis/

12 https://www.simple-scada.com/downloads/Simple-Scada_Manual.pdf

13 https://portal.tpu.ru/SHARED/s/SMIKE/Uchebnaya/Tab3/MasterOPC.pdf

14 Anaconda. Data science technology for groundbreaking research.a competitive edge. A better world.human sensemaking. URL: www.anaconda.com

15 Yadav E. S., Indiran T., Priya S. S., Fedele G. Parameter Estimation and an Extended Predictive-Based Tuning Method for a Lab-Scale Distillation Column. *ACS Omega*, 2019. Vol. 4, No. 25. P. 21230–21241.

16 https://www.novusautomation.com/downloads/Arquivos/manual_txmini-m12-485_transmitter_v10x_j_en.pdf

17 https://a-tcontrols.com/Documents/291/IOM08083.pdf

ДОДАТОК А. Значення ModBus регістрів давачей і плат керування двигунами

Таблиця А.1 – Регістри давача TXTMINI-M12-485

Номер	Зміст	Приклад нормального значення
00	Серійний номер 1	60002
01	Серійний номер 2	30001
02	Номер релізу програмного забезпечення	100
03	Номер моделі	10
05	Значення температури, помножене на 10	700
07	0 – швидкість 1200 бод 1 – швидкість 2400 бод 2 – швидкість 4800 бод 3 – швидкість 9600 бод 4 – швидкість 19200 бод 5 – швидкість 38400 бод 6 – швидкість 57600 бод	3
08	0 – немає контролю чітності 1 – контроль ODD 2 – контроль EVEN	0
09	Номер адреси давача в мережі	1
10	0 – значення показу в °C 1 – значення показу в °F	0
11	0 – є помилка 1 – немає помилки	0
12	Резерв	0
13	Резерв	0
22	Час фільтрації показу, с	5

Таблиця Б.1 – Регістри плати TMB1-RTI

Номер	Зміст	Приклад нормального значення
00	0-немає руху 64-рух за часовою стрілкою 128-рух проти часової стрілки	0
01	Час роботи плати в год (верхній байт)	10
02	Час роботи плати в год (нижній байт)	0
03	Час роботи двигуна в год (верхній байт)	10
05	Час роботи двигуна год (нижній байт)	0
07	Кількість включень (разів)	10
08	Поточне значення регулюючого органу	500
10	Бажане значення регулюючого органу	500
11	Таймаут переміщення	60
12	Таймаут відсутності руху	300
13	Чутливість давача положення, в 0.1%	5

ДОДАТОК Б. Python3-програма симулятора мережі слейвів і технологічного процесу зі збуреннями

```
import pkg_resources
import argparse
import logging
import os
import signal
import struct
import sys
import time
import traceback
from threading import Thread
import configparser
import modbus_tk
import modbus_tk.defines as cst
from modbus_tk import modbus_rtu
from serial import Serial
from datetime import datetime
import numpy as np
from math import floor,ceil
blk = [[None] * 9 for i in range(256)]
import curses
import time

comport=None
start_time = datetime.now()
app_version=0.1
y_simulation = np.zeros((2, 1))
u_simulation = np.zeros((2, 1))
u1_prev = 0
u2_prev = 0

def get_io():
    import random
    global value1, value2
```

```
    value1 = random.randint(1, 30)
    value2 = random.randint(1, 30)

logging.basicConfig(filename="logfile.txt", format='%(asctime)s
(%(levelname)s) %(message)s', level=logging.DEBUG,
                    datefmt='%d.%m.%Y %H:%M:%S')
LOGGER = logging.getLogger(__name__)
LOGGER.addHandler(logging.StreamHandler())
LOGGER.setLevel(logging.WARNING)

thread = None
sim = None

def init_sim():
    global thread
    global config
    global sim

    if config.getboolean('server', 'debug') :
        LOGGER.debug("Running in debug mode")
        return

    if sim is None:
        LOGGER.debug("Sim was not setup so configuring sim")

def add_temperature_sensor(modbus_address, default_val):
    LOGGER.debug("Creating sensor for address %d with default val
%f" %(modbus_address,default_val))
    global server, comport, config

    server.add_slave(modbus_address)
    sensor = server.get_slave(modbus_address)
    sensor.add_block('h', modbus_tk.defines.HOLDING_REGISTERS,
                     0, 23)
    sensor.set_values('h', 0, 60000+modbus_address)  # SN1
    sensor.set_values('h', 1, 30000+modbus_address)  # SN2
```

```
sensor.set_values('h', 2, 100)  # release software
sensor.set_values('h', 3, 12)  # model
sensor.set_values('h', 4, 0)  # N/A
sensor.set_values('h', 5, default_val*10)  # temperature*10
sensor.set_values('h', 6, 0)  # error

v = 0
if comport.baudrate == 1200:
    v = 0
elif comport.baudrate == 2400:
    v = 1
elif comport.baudrate == 4800:
    v = 2
elif comport.baudrate == 9600:
    v = 3
elif comport.baudrate == 19200:
    v = 4
elif comport.baudrate == 38400:
    v = 5
elif comport.baudrate == 57600:
    v = 6
else:
    v = 7

sensor.set_values('h', 7, v)  # 3-9600,4-19200(baudrate)

v = 0
if config.parity == PARITY_NONE:
    v = 0
elif config.parity == PARITY_ODD:
    v = 1
elif config.parity == PARITY_EVEN:
    v = 2

sensor.set_values('h', 8, v)  # parity(0-none, 1-odd, 2-even)
```

```
    sensor.set_values('h', 9, modbus_address)  # modbus address
    sensor.set_values('h', 10, 0)  # temperature unit(0-*C,1-F)
    sensor.set_values('h', 11, 0)  # error value
    for i in range(12, 21,1):
        sensor.set_values('h', i, 0)  # error value

    sensor.set_values('h', 22, 5)  # filter seconds
    return sensor


def add_actuator(modbus_address, default_val):
    global server, comport, config

    server.add_slave(modbus_address)
    sensor = server.get_slave(modbus_address)
    sensor.add_block('h', modbus_tk.defines.HOLDING_REGISTERS,
                     0, 16)
    sensor.set_values('h', 0, 0)  # 0-not moving, 64-CW,128-CCW
    sensor.set_values('h', 1, 10)  # version
    sensor.set_values('h', 2, 0)  # hours online
    sensor.set_values('h', 5, 0)  # motor moving time hr
    sensor.set_values('h', 7, 10)  # motor starts (power ons)
    sensor.set_values('h', 8, default_val*10)# current_pos (read)
    sensor.set_values('h', 10, default_val*10)  # comd pos (write)
    sensor.set_values('h', 11, 60)  # Travel timeout
    sensor.set_values('h', 12, 300)  # Non-movement Timeout
    sensor.set_values('h', 13, 5)  # Sensitivity
    sensor.set_values('h', 14, 1000)  # Commun Timeout
    sensor.set_values('h', 15, 500)  # Falut position

    return sensor

def signal_handler(signm, frame):
    global sim
    LOGGER.info('Got Signal %s, exiting now.' % (str(signm)))
    sim.close()
```

```
    log.info('Stopped distillation process modbus simulator.')
    sys.exit(0)

def load_config(args):
    config = configparser.ConfigParser(allow_no_value=True)
    config.read(args.config)

    if args.port:
        config.port = args.port
    if args.parity:
        config.parity = args.parity
    if args.baud:
        config.baud = args.baud
    if args.com:
        config.com = args.com
    return config

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('-P', '--port', type=int, default=9000,
help='IP port')
    parser.add_argument('-p', '--parity', type=str,
choices=('even','odd','mark','space','none'), default='none',
help='modbus over serial parity')
    parser.add_argument('-b', '--baud', type=int, default=9600,
help='baud rate for modbus')
    parser.add_argument('-c', '--config', type=str,
default='server.conf', help='modbus simulator configuration file')
    parser.add_argument('-m', '--com', type=str, default='COM4',
help='serial port on which to sim')
    parser.add_argument('-x', '--xonxoff', type=bool,
default='False', help='Program control (XON/XOFF), default false')

    args = parser.parse_args()
    return args
```

```
class ServerExit(Exception):
    pass


def service_shutdown(signum, frame):
    raise ServerExit


class SimulationThread(Thread):
    def __init__(self, name):
        Thread.__init__(self)
        self.name = name
        self.__n_states1 = 9#63
        self.__n_states2 = 6
        self.__n_inputs = 2
        self.period = 1000
        self.f_coeff = 100


        self.__Ap = np.mat(
            '[0.995000010442747   0    0.0312500000000000  0    0
     0    0    0    0;0  0.990642613827815   0    0    0    0    0
     0.0312500000000000  0;0  0    0    1    0    0    0    0    0;0
     0    0    0    1    0    0    0    0;0  0    0    0    0    1
     0    0    0;0  0    0    0    0    0    1    0    0;0  0    0
     0    0    0    0    0    0;0  0    0    0    0    0    0    0
     1;0  0    0    0    0    0    0    0    0]')
        self.__Bp = np.mat(
            '[0     0;0  0;0  0;0  0;0  0;0  0;1  0;0  0;0  1]')
        self.__Cp = np.mat(
            '[-0.0224659033253329 0    -0.000187810958864338    0
     0    0    0    0    0;0  0.0540782061662931  0    0    0    0
     0    0    0]')
        self.__xp = np.zeros((self.__n_states1, 1))
        self.__xf = np.zeros((self.__n_states2, 1))
```

```
        self.__xvlv1 = 0
        self.__xvlv2 = 0
        self.u = np.zeros((2, 1))
        self.__dt = 0.18
        self.__cur_step=1
        self.cur_step=1
        self.failure_counter=0

    def simStep(self):
        global config, app_version, blocks, y_simulation, sensor1,
sensor2, actuator1, actuator2, u1_prev, u2_prev
        next_time = time.time() + self.__dt
        self.dt = self.__dt
        cnt1=0
        cnt2=0
        while True:
            time.sleep(max(0, next_time - time.time()))
            try:
                u1 = int(''.join(map(str, actuator1.get_values('h',
10))))/10-50
                u2 = int(''.join(map(str, actuator2.get_values('h',
10))))/10-50

                if u1<-50:
                    u1=-50
                if u2<-20:
                    u2=-20
                LOGGER.debug("Got u1=%f,u2=%f" % (u1,u2))
                LOGGER.debug("Counting u1=%f,u2=%f" % (u1, u2))

                if u1>50:
                    u1=50
                if u2>50:
                    u2=50

                if u1>49.5 and u2>=49.5:
```

```
                self.failure_counter += 1
            else:
                self.failure_counter = 0

            if round((u1-self.__xvlv1),1) >= 0.3:
                self.__xvlv1 += 0.3
                actuator1.set_values('h', 0, 64)
            elif round((u1-self.__xvlv1),1) <= -0.3:
                self.__xvlv1 -= 0.3
                actuator1.set_values('h', 0, 128)
            elif round((u1-self.__xvlv1),1) != 0:
                self.__xvlv1 += round(u1-self.__xvlv1,1)
                if round((u1-self.__xvlv1),1)>0:
                    actuator1.set_values('h', 0, 64)
                else:
                    actuator1.set_values('h', 0, 128)
            elif round((u1 - self.__xvlv1), 1) == 0:
                cnt1=cnt1+1
                if cnt1>13:
                    actuator1.set_values('h', 0, 0)
                    cnt1=0


            actuator1.set_values('h', 8, int((self.__xvlv1 +
50) * 10))

            if round((u2-self.__xvlv2),1) >= 0.3:
                self.__xvlv2 += 0.3
                actuator2.set_values('h', 0, 64)
            elif round((u2-self.__xvlv2),1) <= -0.3:
                self.__xvlv2 -= 0.3
                actuator2.set_values('h', 0, 128)
            elif round((u2-self.__xvlv2),1) != 0:
                self.__xvlv2 += round(u2-self.__xvlv2,1)
                if round((u2-self.__xvlv2),1)>0:
                    actuator2.set_values('h', 0, 64)
```

```
                else:
                    actuator2.set_values('h', 0, 128)
            elif round((u2 - self.__xvlv2), 1) == 0:
                cnt2=cnt2+1
                if cnt2>13:
                    actuator2.set_values('h', 0, 0)
                    cnt2=0


            actuator2.set_values('h', 8, int((self.__xvlv2 +
50) * 10))

            u1_prev = u1
            u2_prev = u2

            self.u[0][0] = self.__xvlv1
            self.u[1][0] = self.__xvlv2

            self.__cur_step += 1
            self.cur_step = self.__cur_step


            th = floor(self.__cur_step / self.period)
            ic = self.__cur_step - th * self.period
            tc = 0.5

            if 0 <= floor(ic) <= 99:
                tc = 0
            elif 99 < floor(ic) <= 199:
                tc = 0.5
            elif 199 < floor(ic) <= 399:
                tc = 1
            elif 399 < floor(ic) <= 699:
                tc = -1
            elif 699 < floor(ic) <= 899:
                tc = -0.5
```

```
                elif 899 < floor(ic) <= 999:
                    tc = 0#-0.5
                f = np.zeros((2, 1))
                f=[0][0] = tc * 0.5
                f=[1][0] = tc * 0.7
                self.__xp = self.__Ap.dot(self.__xp) +
self.__Bp.dot(self.u+f)
                self.__yp = self.__Cp.dot(self.__xp)

                self.y = self.__yp
                y_simulation = self.y
                u_simulation = self.u

                sensor1.set_values('h', 5, 500+ceil(self.y[0][0] *
10))  # temperature*10
                sensor2.set_values('h', 5, 700+ceil(self.y[1][0] *
10))  # temperature*10

            except Exception:
                traceback.print_exc()
                quit()
            next_time += (time.time() - next_time) // self.__dt *
self.__dt + self.__dt

    def run(self):
        global config, app_version, blocks, y_simulation
        self.simStep()


def main():
    global config
    args = parse_args()
    config = load_config(args)

if __name__ == "__main__":
    signal.signal(signal.SIGTERM, service_shutdown)
```

```
    signal.signal(signal.SIGINT, service_shutdown)

    main()

    try:
        PARITY_NONE, PARITY_EVEN, PARITY_ODD, PARITY_MARK,
PARITY_SPACE = 'N', 'E', 'O', 'M', 'S'
        STOPBITS_ONE, STOPBITS_ONE_POINT_FIVE, STOPBITS_TWO = (1,
1.5, 2)
        FIVEBITS, SIXBITS, SEVENBITS, EIGHTBITS = (5, 6, 7, 8)

        PARITY_NAMES = {
            PARITY_NONE: 'None',
            PARITY_EVEN: 'Even',
            PARITY_ODD: 'Odd',
            PARITY_MARK: 'Mark',
            PARITY_SPACE: 'Space',
        }


        comport = Serial(config.get('modbus', 'com'))
        if config.get('modbus', 'stopbits')=='1':
            comport.stopbits = STOPBITS_ONE
        elif config.get('modbus', 'stopbits') == '1.5':
            comport.stopbits = STOPBITS_ONE_POINT_FIVE
        elif config.get('modbus', 'stopbits') == '2':
            comport.stopbits = STOPBITS_TWO
        else:
            print ("Error: stopbits value must be 1, 1.5 or 2")
            quit ()

        if config.get('modbus', 'parity') == 'none':
            config.parity = PARITY_NONE
        elif config.get('modbus', 'parity') == 'even':
            config.parity = PARITY_EVEN
        elif config.get('modbus', 'parity') == 'odd':
```

```
            config.parity = PARITY_ODD
        else:
            print("Error: parity value must be none, even, odd,
mark or space")
            quit()

        BAUDRATES = (1200, 2400, 4800, 9600, 19200, 38400, 57600,
115200)

        if int(config.get('modbus', 'baud')) not in BAUDRATES:
            bstr = ','.join(str(i) for i in BAUDRATES)
            print("Use only standard baudrates: " + bstr)
            quit()
        else:
            comport.baudrate = int(config.get('modbus', 'baud'))


        if config.get('modbus', 'xonxoff') == "True":
            comport.xonxoff = 1
        else:
            comport.xonxoff = 0


        server = modbus_rtu.RtuServer(comport)
        server.start()
        sensor1 = add_temperature_sensor (1,50)
        sensor2 = add_temperature_sensor (2,70)
        actuator1 = add_actuator (3, 50)
        actuator2 = add_actuator (4, 50)


        st = SimulationThread("simulation")
        st.start()

        bar = '▉'
        stdscr = curses.initscr()
```

```
        height, width = stdscr.getmaxyx()
        curses.start_color()
        curses.init_pair(1, curses.COLOR_RED, curses.COLOR_WHITE)
        curses.init_pair(2, curses.COLOR_BLACK,
curses.COLOR_YELLOW)
        curses.init_pair(3, curses.COLOR_RED, curses.COLOR_WHITE)
        curses.init_pair(4, curses.COLOR_GREEN, curses.COLOR_BLACK)

        stdscr.addstr(1, 1, " " * (width - 2),
curses.color_pair(1))
        stdscr.addstr(1, 15, "Distillation Column ModBus Simulator
(written by Zozulya A.)", curses.color_pair(1))
        stdscr.hline(2, 1, "_", width)
        stdscr.addstr(height - 1, 1, " " * (width - 2),
curses.color_pair(1))
        stdscr.addstr(height - 1, 5, "Hit q to quit",
curses.color_pair(1))

        stdscr.addstr(4, 1, "Temperature Top :")
        stdscr.addstr(4, 40, "Temperature Btm :")

        stdscr.addstr(7, 1, "Reflux VLV :")
        stdscr.addstr(7, 40, "Steam VLV :")

        win1 = curses.newwin(7, width-2, 5, 1)
        win2 = curses.newwin(3, 32, 7, 15)

        k = 0
        while (k != ord('q')):
            win1.clear()
            win1.border(1)

            value1 = float(y_simulation[0][0] + 50)
            value2 = float(y_simulation[1][0] + 70)
```

```
            u1 = int(''.join(map(str, actuator1.get_values('h',
10))))/10
            u2 = int(''.join(map(str, actuator2.get_values('h',
10))))/10
            value3 = float(u1)
            value4 = float(u2)

            value1a = round(float(value1),3)
            stdscr.addstr(5, 1,  str(value1a)  + " Deg C ",
curses.A_BOLD)
            value2a = round(float(value2), 3)
            stdscr.addstr(5, 40, str(value2a) + " Deg C ",
curses.A_BOLD)

            stdscr.addstr(8, 1, str(value3) + " % ", curses.A_BOLD)
            stdscr.addstr(8, 40, str(value4) + " % ",
curses.A_BOLD)

            stdscr.addstr(11, 5, "S1", curses.A_BOLD)
            stdscr.addstr(11, 13, "S2", curses.A_BOLD)
            stdscr.addstr(11, 26, "M1", curses.A_BOLD)
            stdscr.addstr(11, 34, "M2", curses.A_BOLD)


regs=["00","01","02","03","05","07","08","09","10","11","12","13","
22"]
            i=0
            for r in range(12,25):
                stdscr.addstr(r, 1, regs[i], curses.A_NORMAL)
                v = " " + str(int(''.join(map(str,
sensor1.get_values('h', int(regs[i])))))) + " "
                v=v+(6-len(v))*" "
                if i==4:
                    stdscr.addstr(r, 4, v, curses.color_pair(3))
                elif i==8 or i==12:
                    stdscr.addstr(r, 4, v, curses.color_pair(4))
```

```
                else:
                    stdscr.addstr(r, 4, v, curses.A_NORMAL)
                i=i+1

            i=0
            for r in range(12,25):
                v = " " + str(int(''.join(map(str,
sensor2.get_values('h', int(regs[i])))))) + " "
                v = v + (6-len(v)) * " "
                if i==4:
                    stdscr.addstr(r, 12, v, curses.color_pair(3))
                elif i==8 or i==12:
                    stdscr.addstr(r, 12, v, curses.color_pair(4))
                else:
                    stdscr.addstr(r, 12, v, curses.A_NORMAL)
                i=i+1

            regs = ["00", "01", "02", "03", "05", "07", "08", "10",
"11", "12", "13", "14", "15"]

            i = 0
            for r in range(12, 25):
                stdscr.addstr(r, 23, regs[i], curses.A_NORMAL)
                v = str(int(''.join(map(str,
actuator1.get_values('h', int(regs[i]))))))
                v = v + (6-len(v)) * " "

                if i==6:
                    stdscr.addstr(r, 26, v, curses.color_pair(3))
                elif i==7:
                    stdscr.addstr(r, 26, v, curses.color_pair(2))
                elif i>7:
                    stdscr.addstr(r, 26, v, curses.color_pair(4))
                else:
                    stdscr.addstr(r, 26, v, curses.A_NORMAL)
                i = i + 1
```

```
            i = 0
            for r in range(12, 25):
                v = str(int(''.join(map(str,
actuator2.get_values('h', int(regs[i]))))))
                v = v + (6-len(v)) * " "
                if i == 6:
                    stdscr.addstr(r, 34, v, curses.color_pair(3))
                elif i == 7:
                    stdscr.addstr(r, 34, v, curses.color_pair(2))
                elif i > 7:
                    stdscr.addstr(r, 34, v, curses.color_pair(4))
                else:
                    stdscr.addstr(r, 34, v, curses.A_NORMAL)
                i = i + 1

            time_elapsed = datetime.now() - start_time
            v = "Version: %.2f, Uptime: %s"
%(app_version,format(time_elapsed))
            stdscr.addstr(30, 1, v, curses.A_NORMAL)
            v = "Port:%s, Baud:%s, Parity: %s, XonXoff: %s,
StopBits: %s" % (config.get('modbus', 'com'),config.get('modbus',
'baud'),config.get('modbus', 'parity'),config.get('modbus',
'xonxoff'),config.get('modbus', 'stopbits'))
            stdscr.addstr(31, 1, v, curses.A_NORMAL)
            v = "Legend: "
            stdscr.addstr(28, 1, v, curses.A_NORMAL)
            v = "Primary RW "
            stdscr.addstr(28, 9, v, curses.color_pair(2))
            v = "Primary R "
            stdscr.addstr(28, 22, v, curses.color_pair(3))
            v = " Secondary RW "
            stdscr.addstr(28, 34, v, curses.color_pair(4))

            stop_time=200
            if st.failure_counter == 0:
```

```
                stdscr.addstr(26, 1, "Column is operating in normal
mode    ", curses.A_BOLD)
            elif (stop_time-st.failure_counter*st.dt)<=1:
                stdscr.addstr(26, 1, "Column is flooded! Stopping
simulation." % (stop_time - st.failure_counter * st.dt),
                                curses.A_BOLD)
                stdscr.refresh()
                for beep in range(0,10):
                    print('\a')
                    pause(1)
                curses.endwin()
Serial.close(comport)
                os.kill(os.getpid(), signal.SIGINT)
                quit()
            else:
                stdscr.addstr(26, 1, "Column failure in %d sec
" % (stop_time-st.failure_counter*st.dt), curses.A_BLINK)

            stdscr.refresh()
            time.sleep(1)
            stdscr.nodelay(1)
            k = stdscr.getch()
        curses.endwin()
        Serial.close(comport)
        os.kill(os.getpid(), signal.SIGINT)
        quit()


    except Exception:
        traceback.print_exc()
        quit()

    except ServerExit:

        print('Server finished')
        Serial.close(comport)
```

```
os.kill(os.getpid(), signal.SIGINT)
quit()
```

ДОДАТОК В. Програмний код скриптів Lua для підміненого OPC-серверу

Увага: оскільки сервер не має глобального простору і глобальних змінних, то деякий код буде повторюватись.

Пристрій Server.DC.TemperatureTop

```
function OnInit()
add_or_update_data_in_file('y1', 50, 'c:\\vals.txt')
add_or_update_data_in_file('tm1', 0, 'c:\\vals.txt')
add_or_update_data_in_file('x1', 0, 'c:\\vals.txt')
add_or_update_data_in_file('ur1', -0.01, 'c:\\vals.txt')
end

function trim1(s)
   return (s:gsub("^%s*(.-)%s*$", "%1"))
end

function read_data_in_file(key1, filespec)
  key1=trim1(key1);
  local file = assert(io.open(filespec))
    for line in file:lines() do
        key, value = string.match(line,"(.-)=(.-)$");
        key=trim1(key);
        value=trim1(value);

        if key==key1 then
            return value
        end
    end
    file:close()
    return nil;
end

function round(num, numDecimalPlaces)
  local mult = 10^(numDecimalPlaces or 0)
  return math.floor(num * mult + 0.5) / mult
end

function add_or_update_data_in_file(key, new_value, filespec)
  local file = assert(io.open(filespec))
  local content = file:read'*a'
  file:close()
  content, cnt = content:gsub(
    '(%f[%C]%s*'..key..'%s*=[ \t]*)[^\r\n]*',
    '%1'..new_value, 1)
  if cnt == 0 then
    content = key..' = '..new_value..'\n'..content
  end
  file = assert(io.open(filespec, 'w'))
  file:write(content)
  file:close()
end
```

```
function OnBeforeReading()
    file = io.open('c:\\hacked.txt', "r")
    io.input(file)
    local ret=(io.read())
    io.close(file)

    if trim1(ret)=="1" then
        y1=tonumber(read_data_in_file('y1', 'c:\\vals.txt'))
        math.randomseed(os.time())
        tm1=read_data_in_file('tm1', 'c:\\vals.txt')
        tm1=tm1+1
        add_or_update_data_in_file('tm1', tm1, 'c:\\vals.txt')
        ur1=tonumber(read_data_in_file('ur1', 'c:\\vals.txt'))
        if math.fmod(tm1,10)==0 then
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            r=math.random()
            if ur1>0.2 then
                ur1=ur1-0.1;
            elseif ur1<-0.2 then
                ur1=ur1+0.1;
            elseif r>0.5 then
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
                ur1=ur1+math.random()*-0.15;
            else
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                ur1=ur1+math.random()*0.15;
            end
            add_or_update_data_in_file('ur1', ur1, 'c:\\vals.txt')
        end
        k=1;
        T=3;
        yf=k*(1-math.exp(-tm1/T))*(ur1+(y1-50))
        yf=round(yf,1);
        server.WriteTag("DC.Temperature
Top.Measurement",50+yf,OPC_QUALITY_GOOD);
    else
        err,w =
modbus.ReadHoldingRegistersAsUInt16(5,2,true,"56781234");
        server.WriteTag("DC.Temperature
Top.Measurement",w[1]/10,OPC_QUALITY_GOOD);
        add_or_update_data_in_file('y1', w[1]/10, 'c:\\vals.txt')
        add_or_update_data_in_file('tm1', 0, 'c:\\vals.txt')
    end
```

```
end
-- Processing
function OnAfterReading()
    TV,TQ,TS = server.ReadTag("DC.Temperature Top.Measurement");
    _G.y1=TV
end
```

Пристрій Server.DC.TemperatureBottom

```
function OnInit()
add_or_update_data_in_file('y2', 70, 'c:\\vals.txt')
add_or_update_data_in_file('tm2', 0, 'c:\\vals.txt')
add_or_update_data_in_file('x2', 0, 'c:\\vals.txt')
add_or_update_data_in_file('ur2', -0.01, 'c:\\vals.txt')
end

function trim1(s)
   return (s:gsub("^%s*(.-)%s*$", "%1"))
end

function read_data_in_file(key1, filespec)
  key1=trim1(key1);
  local file = assert(io.open(filespec))
    for line in file:lines() do
        key, value = string.match(line,"(.-)=(.-)$");
        key=trim1(key);
        value=trim1(value);

        if key==key1 then
            return value
        end
    end
    file:close()
    return nil;
end

function round(num, numDecimalPlaces)
  local mult = 10^(numDecimalPlaces or 0)
```

```
  return math.floor(num * mult + 0.5) / mult
end

function add_or_update_data_in_file(key, new_value, filespec)
  local file = assert(io.open(filespec))
  local content = file:read'*a'
  file:close()

  content, cnt = content:gsub(
    '(%f[%C]%s*'..key..'%s*=[ \t]*)[^\r\n]*',
    '%1'..new_value, 1)
  if cnt == 0 then
    content = key..' = '..new_value..'\n'..content
  end
  file = assert(io.open(filespec, 'w'))
  file:write(content)
  file:close()
end

function OnBeforeReading()
    file = io.open('c:\\hacked.txt', "r")
    io.input(file)
    local ret=(io.read())
    io.close(file)

    if trim1(ret)=="1" then
        y2=tonumber(read_data_in_file('y2', 'c:\\vals.txt'))
        math.randomseed(os.time())
        tm2=read_data_in_file('tm2', 'c:\\vals.txt')
        tm2=tm2+1
        add_or_update_data_in_file('tm2', tm2, 'c:\\vals.txt')
        ur2=tonumber(read_data_in_file('ur2', 'c:\\vals.txt'))
        if math.fmod(tm2,10)==0 then
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
```

```
            math.random(); math.random(); math.random()
            r=math.random()
            if ur2>0.2 then
                ur2=ur2-0.1;
            elseif ur2<-0.2 then
                ur2=ur2+0.1;
            elseif r>0.5 then
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
            math.random(); math.random(); math.random()
                ur2=ur2+math.random()*-0.15;
            else
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                math.random(); math.random(); math.random()
                ur2=ur2+math.random()*0.15;
            end
            add_or_update_data_in_file('ur2', ur2, 'c:\\vals.txt')
        end
        k=1;
        T=3;
        yf=k*(1-math.exp(-tm2/T))*(ur2+(y2-70))
        yf=round(yf,1);
        server.WriteTag("DC.Temperature
Bottom.Measurement",70+yf,OPC_QUALITY_GOOD);
    else
        err,w =
modbus.ReadHoldingRegistersAsUInt16(5,2,true,"56781234");
        server.WriteTag("DC.Temperature
Bottom.Measurement",w[1]/10,OPC_QUALITY_GOOD);
        add_or_update_data_in_file('y2', w[1]/10, 'c:\\vals.txt')
        add_or_update_data_in_file('tm2', 0, 'c:\\vals.txt')
    end
```

```
end
```

Пристрій Server.DC.Reflux

```
function OnInit()
       server.WriteTag("DC.Reflux.MV",50.0,OPC_QUALITY_GOOD);
       _G.time1=os.time(datetime);
end

function trim1(s)
   return (s:gsub("^%s*(.-)%s*$", "%1"))
end

function int2bin(n)
  local result = ''
  while n ~= 0 do
    if n % 2 == 0 then
      result = '0' .. result
    else
      result = '1' .. result
    end
    n = math.floor(n / 2)
  end
  return result
end

function OnAfterReading()
    local w={}

    local err

    file = io.open('c:\\hacked.txt', "r")
    io.input(file)
    local ret=(io.read())

    io.close(file)
```

```
if trim1(ret) == "1" then
    num=1000
else
    num,TQ,TS = server.ReadTag("DC.Reflux.MV");
    num=num*10
    if num>1000 then
        num=1000
        server.WriteTag("DC.Reflux.MV",100.0,OPC_QUALITY_GOOD);
    end

    if num<300 then
        num=300
        server.WriteTag("DC.Reflux.MV",30.0,OPC_QUALITY_GOOD);
    end
end
bits=int2bin(num)
toadd=16-string.len(bits)
res1=(string.format( "%s%s", string.rep( "0", toadd ),bits))
res=string.reverse(res1)
n = {}
for i=0,15,1 do
     n[i] = (tonumber(string.sub( res, i+1,i+1)))
end

byte1=n[0]
for i=1,7 do
    byte1 = byte1 + (2*n[i])^(i)
end
byte2=n[8]
for i=9,15 do
    byte2 = byte2 + (2*n[i])^(i-8)
end
```

```
    err =
modbus.WriteHoldingRegistersAsString(10,2,string.format("%s%s",stri
ng.char(byte2),string.char(byte1)),true);


    if trim1(ret) == "1" then
        server.WriteTag("DC.Reflux.Moving",0,OPC_QUALITY_GOOD);
    else
        seconds = os.time(datetime);

        err,w =
modbus.ReadHoldingRegistersAsUInt16(0,2,true,"56781234");
        if w[1]==0 and (_G.time1-seconds)>1 then
            local skip=1
        else

server.WriteTag("DC.Reflux.Moving",w[1],OPC_QUALITY_GOOD);
        end
        _G.time1=seconds
    end
    if trim1(ret) == "1" then
        num,TQ,TS = server.ReadTag("DC.Reflux.MV");
        server.WriteTag("DC.Reflux.MVnow",num,OPC_QUALITY_GOOD);
    else
        err,w =
modbus.ReadHoldingRegistersAsUInt16(8,2,true,"56781234");

server.WriteTag("DC.Reflux.MVnow",w[1]/10,OPC_QUALITY_GOOD);
    end
end
```

Пристрій Server.DC.Steam

```
function OnInit()
       server.WriteTag("DC.Steam.MV",50.0,OPC_QUALITY_GOOD);
       _G.time2=os.time(datetime);
```

```
end

function trim1(s)
   return (s:gsub("^%s*(.-)%s*$", "%1"))
end
function int2bin(n)
  local result = ''
  while n ~= 0 do
    if n % 2 == 0 then
      result = '0' .. result
    else
      result = '1' .. result
    end
    n = math.floor(n / 2)
  end
  return result
end

function OnAfterReading()
    local w={}

    local err

    file = io.open('c:\\hacked.txt', "r")
    io.input(file)
    local ret=(io.read())

    io.close(file)


    if  trim1(ret) == "1" then
        num=1000
    else
        num,TQ,TS = server.ReadTag("DC.Steam.MV");
        num=num*10
        if num>1000 then
```

```
            num=1000
            server.WriteTag("DC.Steam.MV",100.0,OPC_QUALITY_GOOD);
        end

        if num<300 then
            num=300
            server.WriteTag("DC.Steam.MV",30.0,OPC_QUALITY_GOOD);
        end
    end
    bits=int2bin(num)
    toadd=16-string.len(bits)
    res1=(string.format( "%s%s", string.rep( "0", toadd ),bits))
    res=string.reverse(res1)
    n = {}
    for i=0,15,1 do
         n[i] = (tonumber(string.sub( res, i+1,i+1)))
    end

    byte1=n[0]
    for i=1,7 do
        byte1 = byte1 + (2*n[i])^(i)
    end
    byte2=n[8]
    for i=9,15 do
        byte2 = byte2 + (2*n[i])^(i-8)
    end

    err =
modbus.WriteHoldingRegistersAsString(10,2,string.format("%s%s",stri
ng.char(byte2),string.char(byte1)),true);


    if trim1(ret) == "1" then
        server.WriteTag("DC.Steam.Moving",0,OPC_QUALITY_GOOD);
    else
        seconds = os.time(datetime);
```

```
        err,w =
modbus.ReadHoldingRegistersAsUInt16(0,2,true,"56781234");
        if w[1]==0 and (_G.time2-seconds)>1 then
            local skip=1
        else

server.WriteTag("DC.Steam.Moving",w[1],OPC_QUALITY_GOOD);
        end
        _G.time2=seconds
    end
    if trim1(ret) == "1" then
        num,TQ,TS = server.ReadTag("DC.Steam.MV");
        server.WriteTag("DC.Steam.MVnow",num,OPC_QUALITY_GOOD);
    else
        err,w =
modbus.ReadHoldingRegistersAsUInt16(8,2,true,"56781234");
        server.WriteTag("DC.Steam.MVnow",w[1]/10,OPC_QUALITY_GOOD);
    end
end
```

ДОДАТОК Г. Перелік тегів SCADA

Ім'я змінної	Тип даних	Тип тега	Тег в OPC сервері	Призначення
state_steam	Integer	внутріш.		Індикація ненормативного значення витрати пари
state_reflux	Integer	внутріш.		Індикація ненормативного значення витрати флегми
state_ctop	Integer	внутріш.		Індикація ненормативного значення темп. зверху
state_cbtm	Integer	внутріш.		Індикація ненормативного значення темп. знизу
state_reg1	Integer	внутріш.		Індикація порушення роботи регулятора темп. зверху
state_reg2	Integer	внутріш.		Індикація порушення роботи регулятора темп. знизу
state_hacked	Single	внутріш.		Індикація кібератаки
var_sp1	Double	внутріш.		Завдання на температуру зверху
var_sp2	Double	внутріш.		Завлання на температуру знизу
var_y1_normed	Double	внутріш.		Масштабоване до -5..5 значення темп. зверху
var_y2_normed	Double	внутріш.		Масштабоване до -5..5 значення темп. Знизу
var_y1	Single	зовніш.	DC.Temperature Top.Measurement	Виміряне значення темп. зверху
var_y2	Single	зовніш.	DC.Temperature Bottom.Measurement	Виміряне значення темп. Знизу
var_u1	Single	зовніш.	DC.Reflux.MV	Передане значення витрати флегми
var_u2	Single	зовніш.	DC.Steam.MV	Передане значення витрати пари

reg1_e0	Double	внутріш.		Внутрішні змінні регуляторів
reg1_u0	Double	внутріш.		
reg1_init	Double	внутріш.		
reg2_u0	Double	внутріш.		
reg2_init	Double	внутріш.		
var_u1_mvnow	Single	зовніш.	DC.Reflux.MVnow	Виміряне значення положення РО флегми
var_u2_mvnow	Single	зовніш.	DC.Steam.MVnow	Виміряне значення положення РО пари
var_u1_movingto	Single	зовніш.	DC.Reflux.Moving	Напрям руху РО флегми
var_u2_movingto	Single	зовніш.	DC.Steam.Moving	Напрям руху РО пари

ДОДАТОК Д. Програмний код скрипта періодичного виклику для реалізації сигналізації в SCADA мовою Free Pascal

```
procedure lampUpdate (Sender:TM_Control)
procedure indicateVar (stateVar:TM_VARIABLE;
curVariable:TM_VARIABLE; errorValue:Real; alarmErrorValue:Real;
warningErrorValue:Real; deadZoneValue:Real; alarmMessage:String;
warningMessage:String;
normMessage:String);
var
  state_before:Integer;
  state_after:Integer;
  greyColor: Cardinal;
begin
  greyColor  := RGB(224,224,218);
if curVariable.IsGoodQuality then begin
  state_before:=stateVar.Value;
  if alarmErrorValue<=errorValue then  begin
     stateVar.Value:=2;
  end
  else begin
    if errorValue < (alarmErrorValue-deadZoneValue) then  begin
      if errorValue>(warningErrorValue+deadZoneValue) then  begin
        stateVar.Value:=1;
      end
      else  begin
        stateVar.Value:=0;
      end
    end;
  end;
  state_after := stateVar.Value;

  if (state_after<>state_before) then begin
        if stateVar.Value=1 then begin
```

```
            AddMessage(Now,mkWarning,warningMessage+ '
(VAR='+curVariable.Value+')',True,True);
        end;
        if stateVar.Value=2 then begin
            AddMessage(Now,mkAlarm,alarmMessage+ '
(VAR='+curVariable.Value +')',True,True);
        end;
        if stateVar.Value=0 then begin
           AddMessage(Now,mkMessage,  normMessage + '
(VAR='+curVariable.Value + ')',
           True,True);
        end;
  end;
end;
end;
begin

 indicateVar (state_reflux, var_u1, Abs(var_u1.Value-50), 50, 30,
0.3,
 'Витрата флегми аварійна!','Витрата флегми вийшла за межі
регламенту!',
 'Витрата флегми в нормі');
 indicateVar (state_steam, var_u2, Abs(var_u2.Value-50), 50, 30,
0.3,
 'Витрата пари аварійна!','Витрата пари вийшла за межі
регламенту!',
 'Витрата пари в нормі');

  indicateVar (state_ctop, var_y1, Abs(var_y1.Value-50), 5, 2, 0.1,
'Температура верху аварійна!','Температура верху вийшла за межі
регламенту!',
'Температура верху в нормі');
  indicateVar (state_cbtm, var_y2, Abs(var_y2.Value-70), 5, 2, 0.1,
'Температура низу аварійна!','Температура низу вийшла за межі
регламенту!',
'Температура низу в нормі');
```

```
var_y1_normed.Value := var_y1.Value-50+5; //50 nom
var_y2_normed.Value := var_y2.Value-70+5; //80 nom

if   state_reflux.Value = 1 then begin
     ind2_reflux.Visible:=True;  ind2_reflux.Frame:=2;
end;
if   state_reflux.Value = 2 then begin
     ind2_reflux.Visible:=True;  ind2_reflux.Frame:=1;
end;
if   state_reflux.Value = 0 then begin
     ind2_reflux.Visible:=False;
end;


if   state_steam.Value = 1 then begin
     ind2_steam.Visible:=True;  ind2_steam.Frame:=2;
end;
if   state_steam.Value = 2 then begin
     ind2_steam.Visible:=True;  ind2_steam.Frame:=1;
end;
if   state_steam.Value = 0 then begin
     ind2_steam.Visible:=False;
end;


if   state_ctop.Value = 1 then begin
     ind2_ctop.Visible:=True;  ind2_ctop.Frame:=2;
end;
if   state_ctop.Value = 2 then begin
     ind2_ctop.Visible:=True;  ind2_ctop.Frame:=1;
end;
if   state_ctop.Value = 0 then begin
     ind2_ctop.Visible:=False;
end;
```

```
if   state_cbtm.Value = 1 then begin
     ind2_cbtm.Visible:=True;  ind2_cbtm.Frame:=2;
end;
if   state_cbtm.Value = 2 then begin
     ind2_cbtm.Visible:=True;  ind2_cbtm.Frame:=1;
end;
if   state_cbtm.Value = 0 then begin
     ind2_cbtm.Visible:=False;
end;

end.
```

ДОДАТОК Е. Програмний код системи реалізації автоматичного керування з збереження історії керувань

Програмний модуль

```
Unit Glb;
interface
type
vector10 = array [1..10] of Double;
var
u1_history: vector10;
u2_history: vector10;
u1_diff: vector10;
u2_diff: vector10;
u1_movingto: vector10;
u2_movingto: vector10;
u1_mvnow: vector10;
u2_mvnow: vector10;
cnt1:longint;
cnt2:longint;
u1_diff_cnt:Double;
u2_diff_cnt:Double;
implementation

end.
```

Скрипт періодичного виклику

```
procedure HistoryAndControl(Sender: TM_Control);
var
reg1_Kp,reg1_Ki:Double;
reg2_Kp,reg2_Ki:Double;
u1,e1,u2,e2:Double;
i:longint;

begin
i:=0;

{reg1_Kp  := -0.1;
reg1_Ki  := 0;
reg2_Kp  := 0.05;
```

```
reg2_Ki  := 0;}

reg1_Kp  := -4/5;
reg1_Ki  := (-4/5)/30;
reg2_Kp  := 3.5/5;
reg2_Ki  := (3.5/5)/16;

if reg1_init.Value <> 1 then
  begin
  reg1_e2.Value    :=0;
  reg1_e1.Value    :=0;
  reg1_e0.Value    :=0;
  reg1_u2.Value    :=0;
  reg1_u1.Value    :=0;
  reg1_u0.Value    :=0;
  reg1_init.Value  :=1;
  var_sp1.Value:=50;
  var_u1.Value:=50;
  u1:=0;
  e1:=0;

  var_sp2.Value:=70;
  var_u2.Value:=50;
  u2:=0;
  e2:=0;

  for i:=10 downto 1 do
  begin
    u1_history[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u2_history[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u1_diff[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u2_diff[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u1_mvnow[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
```

```
    u2_mvnow[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u2_movingto[i]:=0;
  end;

  for i:=10 downto 1 do
  begin
    u1_movingto[i]:=0;
  end;


  end;

e1 :=   var_sp1.Value-var_y1.Value;
e2 :=   var_sp2.Value-var_y2.Value;


u1:=reg1_u0.Value + reg1_Kp * e1 + reg1_Kp*reg1_Ki*reg1_e0.Value;
if u1>50 then u1:=50;
if u1<-20 then u1:=-20;
reg1_u0.Value:=u1;
reg1_e0.Value:=e1;
if (u1+50)>100 then begin
   var_u1.Value := u1+50;
   end
else if u1<=-30 then begin
   var_u1.Value := 30;
   end
else  begin
  var_u1.Value := u1+50;
  end ;

u2:=reg2_u0.Value + reg2_Kp * e2 + reg2_Kp*reg2_Ki*reg2_e0.Value;
if u2>50 then u2:=50;
if u2<-20 then u2:=-20;
reg2_u0.Value:=u2;
reg2_e0.Value:=e2;
if (u2+50)>100 then begin
   var_u2.Value := u2+50;
   end
else if u2<=-30 then begin
   var_u2.Value := 30;
   end
else  begin
  var_u2.Value := u2+50;
  end ;
end.
```

ДОДАТОК Ж. Програмний код системи кіберзахисту в SCADA-системі

```
procedure CyberSecurity(Sender: TM_Control);
var
i:longint;
begin
for i:=10 downto 2 do
begin
  u1_history[i]:=u1_history[i-1];
end;
u1_history[1]:=var_u1.Value;

for i:=10 downto 2 do
begin
  u1_mvnow[i]:=u1_mvnow[i-1];
end;
u1_mvnow[1]:=var_u1_mvnow.Value;

for i:=10 downto 2 do
begin
  u1_movingto[i]:=u1_movingto[i-1];
end;
u1_movingto[1]:=var_u1_movingto.Value;


for i:=10 downto 2 do
begin
  u1_diff[i]:=u1_history[i]-u1_history[i-1];
end;
u1_diff[1]:=0;

u1_diff_cnt:=u1_diff_cnt+u1_diff[2];

for i:=10 downto 2 do
begin
  u2_mvnow[i]:=u2_mvnow[i-1];
end;
u2_mvnow[1]:=var_u2_mvnow.Value;

for i:=10 downto 2 do
begin
  u2_movingto[i]:=u2_movingto[i-1];
end;
u2_movingto[1]:=var_u2_movingto.Value;


for i:=10 downto 2 do
begin
  u2_history[i]:=u2_history[i-1];
end;
u2_history[1]:=var_u2.Value;
```

```
for i:=10 downto 2 do
begin
  u2_diff[i]:=u2_history[i]-u2_history[i-1];
end;
u2_diff[1]:=0;
u2_diff_cnt:=u2_diff_cnt+u2_diff[2];


for i := 1 to 10 do
begin
  Table1.Columns[1][i].Text  :=FloatToStr(u1_history[i],3);
end;

for i := 1 to 10 do
begin
  Table1.Columns[2][i].Text  :=FloatToStr(u1_diff[i],3);
end;
if Abs(u1_diff_cnt) > 0.1 then
begin
  cnt1:=cnt1+1;
  u1_diff_cnt:=0;
end;

for i := 1 to 10 do
begin
  Table1.Columns[3][i].Text  :=FloatToStr(u1_mvnow[i],3);
end;


for i := 1 to 10 do
begin
  Table1.Columns[4][i].Text  :='';
  Table1.Columns[5][i].Text  :='';
  if u1_movingto[i] = 64  then Table1.Columns[4][i].Text  :='X';
  if u1_movingto[i] = 128 then Table1.Columns[5][i].Text  :='X';
  if u1_movingto[i] <> 0  then cnt1:=0;
end ;


for i := 1 to 10 do
begin
  Table2.Columns[1][i].Text  :=FloatToStr(u2_history[i],3);
end;

for i := 1 to 10 do
begin
  Table2.Columns[2][i].Text  :=FloatToStr(u2_diff[i],3);
end;
if Abs(u2_diff_cnt) > 0.1 then
begin
  cnt2:=cnt2+1;
  u2_diff_cnt:=0;
end;
```

```
for i := 1 to 10 do
begin
  Table2.Columns[3][i].Text  :=FloatToStr(u2_mvnow[i],3);
end;


for i := 1 to 10 do
begin
  Table2.Columns[4][i].Text  :='';
  Table2.Columns[5][i].Text  :='';
  if u2_movingto[i] = 64  then Table2.Columns[4][i].Text  :='X';
  if u2_movingto[i] = 128 then Table2.Columns[5][i].Text  :='X';
  if u2_movingto[i] <> 0  then cnt2:=0;
end ;

fcnt1.Text := IntToStr(cnt1);
fcnt2.Text := IntToStr(cnt2);
ncnt1.Text := FloatToStr(u1_diff_cnt);
ncnt2.Text := FloatToStr(u2_diff_cnt);

if (cnt1>10) or (cnt2>10) then  begin
  ind1_hacked.Color := RGB(255,0,0);
  if state_hacked.Value <> 1 then
    begin
      state_hacked.Value :=1;
      AddMessage(Now,mkAlarm,'Проходить кібератака OPC
серверу',True,True);
      cmdKillOPc.Visible:=True;
    end;
  end
else
  begin
  ind1_hacked.Color := RGB(224,224,218);
    if state_hacked.Value <> 0 then
    begin
      state_hacked.Value :=0;
      AddMessage(Now,mkMessage,  'Кібератака мабуть подолана',
True,True);
      cmdKillOPc.Visible:=False;
    end;
  end;


ind2_hacked.Color := ind1_hacked.Color;

end.
```