Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Іщенко Д.А.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Кислун О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Іщенку Денису Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP*

2. Керівник роботи *Кислун Олег Андрійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 35-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Іщенко Д.А. Дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Метою розробки є дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Об'єктом дослідження є процес платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Предметом дослідження є методи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, інтелектуальний будинок, CWMP

ABSTRACT

Ishchenko D.A. Research and software implementation of the intelligent home network device management platform system based on CWMP technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the platform system of managing network devices of an intelligent house using CWMP technology.

The purpose of the development is the research and software implementation of the intelligent home network device management platform system based on CWMP technology.

The object of the study is the process of the management platform of network devices of the intelligent house using CWMP technology.

The subject of the study is the methods of the platform for managing network devices of an intelligent house using CWMP technology.

Research methods are based on Internet of Things methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the platform system for managing network devices of an intelligent house based on CWMP technology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Builder C++ environment.

Keywords: computer engineering, intelligent building, CWMP

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АРМ	–	автоматизоване робоче місце
АСУ	–	автоматизована система управління
ДБЖ	–	джерело безперебійного живлення
ДКС	–	домашня кабельна мережа
ДУ	–	дистанційне управління
ЕОМ	–	електронно-обчислювальна машина
ЕФ	–	екранна форма
ЗТО	–	звукова трансляція й оповіщення
ІБ	–	інтелектуальний будинок
ІЧ	–	інфрачервоний
ОВК	–	управління опаленням, вентиляцією й кондиціюванням
ОДС	–	оперативна диспетчерська система
ОПС	–	охоронно-пожежна сигналізація
ПДУ	–	пульти дистанційного управління
ПЗ	–	програмне забезпечення
ПЛК	–	програмувальні логічні контролери
ПМО	–	програмно-математичного забезпечення
РК	–	рідкокристалічний
СКК	–	система кабельних комунікацій
ТЗ	–	технічне завдання
ЕІВ	–	європейська інсталяційна шина

ВСТУП

Актуальність теми. «Розумний будинок» або «інтелектуальний будинок» – це централізована автоматизована система управління всіма електричними навантаженнями, інженерними системами й мультимедійним устаткуванням у будинку, що дозволяє досягти нового рівня комфорту, безпеки й енергозбереження.

Можливості системи «розумний будинок» обмежуються тільки Вашою уявою. Ви можете створювати різні сценарії, поєднуючи між собою систему освітлення, опалення, кондиціювання, систему безпеки й відеоспостереження, домашній кінотеатр, систему управління воротами, вікнами, жалюзями. У теж час інженерією легко управляти за допомогою сенсорних панелей і вимикачів, універсального пульта, мобільного телефону.

Система «інтелектуальний будинок» надає наступні послуги:

– Управління освітленням. Ви можете управляти освітленням у будинку й на вулиці, при наявності/відсутності руху, створювати світлові сценарії, ефект присутності під час Вашої відсутності, виключати все освітлення, ідучи з будинку.

– Управління мікрокліматом. Система забезпечує загальне управління приточно-витяжною вентиляцією, кондиціонерами, казанами, системою обігріву, теплими підлогами, теплою покрівлею. Ви можете централізовано регулювати мікроклімат у всьому будинку або індивідуально кожної кімнати.

– Мультирум. Медіацентр. Це система розподілу звуку й відео, що охоплює велику кількість окремих приміщень, від віталень і спалень до кухні, ванною й навіть комори. Мультирум також може забезпечити «ландшафтне озвучування присадибної ділянки під час літньої вечірки.

– Відеоспостереження. Детальний звіт подій, що відбуваються під час вашої відсутності. Особи й дії фіксуються в системі й зберігаються в її пам'яті. Ви одержуєте можливість оперативного перегляду ситуації на будь-якому моніторі або комп'ютері в будь-якій точці миру, використовуючи Інтернет.

– Захист від затоплення й витоку газу. Відключення газу або води (залежно від типу аварійної ситуації), повідомлення у відповідні служби, а також на мобільний телефон замовника.

– Охоронна й пожежна сигналізація. Визначення факту несанкціонованого проникнення на охоронюваний об'єкт або появи ознак пожежі, видачі сигналу тривоги й включення виконавчих пристроїв (світлових і звукових оповіщувачів, реле й т.п.).

– Засоби управління Розумним будинком. Система підтримує всі сучасні доступні засоби управління, починаючи від звичайного вимикача, або пульта дистанційного управління, закінчуючи сенсорною панеллю із графічним інтерфейсом, кишеньковим комп'ютером або мобільним телефоном.

Завдяки системі «розумний будинок» різні підсистеми починають працювати погоджені, інженерне устаткування – самостійно, з'являється поняття «сценарій», коли по натисканню однієї кнопки, відбувається будь-який набір дій.

У результаті Ви одержуєте – комфорт, безпеку й економію електроенергії.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– Дослідження системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– Програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Об’єктом дослідження є процес платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Предметом дослідження є методи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– Розроблено вітчизняний продукт платформи управління мережними пристроями інтелектуального будинку за технологією CWMP, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для управління мережними пристроями інтелектуального будинку за технологією CWMP**.** Мільярди пристроїв з'єднують нас із все новими сервісами й додатками, а машини починають прямо взаємодіяти з іншими машинами. У цьому новому світі як ніколи важливими стають способи управління складним комплексним середовищем.

У галузі «золотим стандартом» віддаленого управління вважається TR-069 – розроблений Broadband Forum протокол CPE WAN Management Protocol (CWMP). Він дозволяє коректно й економічно ефективно віддалено виділяти й звільняти ресурси, здійснювати моніторинг широкого спектра пристроїв і систем, установлених удома або в офісі, управляти ними й контролювати їхню роботу. Відповідно до результатів дослідження Ovum 2015 року, TR-069 одержав широке поширення на всіх континентах: 150 млн. пристроїв управляються за цим протоколом, і їхня кількість швидко росте.

1.2 Область застосування

У цей час TR-069 – світовий стандарт, на який опираються багато галузевих організацій, включаючи 3GPP, ATIS, CCSA, ETSI і HGI. Недавно він одержав офіційну рекомендацію ITU-T Recommendation (G.9980), а в 2010 році був схвалений ETSI як європейський стандарт. TR-069 використовується також в архітектурі ETSI Machine-to-Machine (M2M) як протокол управління пристроями. У листопаді 2012 року Broadband Forum одержав за TR-069 престижну нагороду InfoVision «За видатний внесок у забезпечення широкополосного доступу» (Outstanding Contribution to Broadband Success).

У сучасних системах домашньої автоматизації для запобігання накопичення великої кількості пультів і забезпечення комфортної комунікації з технікою використовуються панелі управління (сенсорні панелі, тач-панелі, touchscreen).

Функції панелей управління

Візуалізація, управління й контроль інтегрованих систем:

- Системи безпеки.
- Опалення або кондиціонерні системи.
- Освітлення або роллети.
- Телефонні системи.
- Користування Інтернетом або електронною поштою.
- Колекції DV/DVD або музики.
- Система домофоного зв'язку.
- Домашній кінотеатр.
- Інформаційні фільтри: прогноз погоди, інформація з дорожніх заторів, біржові новини й курси й т.д.

На ринку технологій комфорту досить велика кількість виробників Панелей управління. Різниця в більшості визначається винятково дизайном панелей.

Панелі управління умовно можна розділити на 2 категорії – що вбудовуються й переносні.

Убудовані сенсорні панелі

Екрани, що вбудовуються в стіну, чутливі до торкання пальця. Мають індивідуальну графіку, можуть показувати відеозображення. Розмір 8", 15", 19", 22".

Переваги: просте, інтуїтивно зрозуміле управління всіма системами. Багата графіка. Українська мова.

Недоліки: постійне місце установки, до якого потрібно підходити.

Переносні сенсорні панелі

Переносні планшети з екраном, чутливим до торкання. Працюють на базі Wi-Fi. Графіка аналогічна убудованим. Зручно використовувати на території, SPA-зоні.

Переваги: можна носити по всьому будинку й ділянці. Просте, інтуїтивно зрозуміле управління всіма системами. Українська мова.

Недоліки: вимагають регулярної підзарядки. Занадто великі (крім IPhone), щоб постійно носити із собою. Висока ймовірність розбити, втратити.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

У даному розділі проведемо огляд найсучасніших і інноваційних пристроїв, які реалізуються в технології інтелектуального будинку.

Датчик, що вчасно попередить про витік газу

Рисунок 2.1 – Зовнішній вигляд датчика, що вчасно попередить про витік газу

На Кікстартері кілька днів назад почалася кампанія в підтримку чергового гаджету для розумного будинку. Так як подібні пристрої зараз у тренді, новинка досить швидко зібрала необхідні кошти й повинна надійти в продаж у листопаді за ціною близько 70 канадських доларів. Kepler є сенсором, що постійно стежить за кількістю метану й вуглекислого газу в приміщенні й попереджуючої про високий рівень забруднення звуковим сигналом. Зрозуміло, дані про состав повітря можна одержувати на смартфон, підключивши його до пристрою. Так само, розуміючи, що в житлових приміщеннях Kepler найчастіше буде встановлюватися на кухнях, розроблювачі додали в нього функцію таймера, що допоможе контролювати час готування їжі.

iControl Piper

Рисунок 2.2 – Зовнішній вигляд iControl Piper

Цей проект був згаданий ледве більше роки тому, тоді він ще називався Blacksumac Piper. Пізніше проект був перекуплений iControl і вийшов на ринок під його брендом. У відмінності від звичайних камер відеоспостереження, дана модель є сьогоденням керуючим центром домашньої системи безпеки й автоматизації.

WiFi-роутер із сенсорним дисплеєм для розумного будинку

Рисунок 2.3 – Зовнішній вигляд WiFi-роутера із сенсорним дисплеєм для розумного будинку

Об'єднання функцій різних пристроїв в одному апарату зустрічається всі частіше й, незважаючи на те, що деякі подібні гібриди виглядають досить дивно, інші виявляються досить корисними. Саме до останнього можна віднести WiFi-роутер Soap із сенсорним екраном. Крім забезпечення бездротового зв'язку, ця новинка має функцію контролю більшості гаджетів, що працюють із нею в одній мережі, а операційна система Android дозволяє створити аналог розумного будинку. При бажанні, управління можна перепризначити на смартфон або планшет, які завжди перебувають під рукою. Серед корисних функцій пристрою так само можна відзначити систему захисту даних і батьківський контроль, які дозволяють обмежити доступ до певних пристроїв, наприклад, ігрової консолі.

На даний момент проект Soap представлений на indiegogo і, у випадку успішного завершення кампанії, придбати роутер можна буде в лютому наступного року за ціною біля $240.

Розумні меблі для маленьких квартир

Рисунок 2.4 – Зовнішній вигляд розумних меблів для маленьких квартир

Схоже, що гуртожиток Массачусетського технологічного інституту має досить маленькі кімнати для студентів. Саме цим можна пояснити проект, що з'явився в його надрах, названий CityHome. Його суть полягає в тому, щоб з максимальною ефективністю використовувати наявний невеликий простір кімнати або квартири-студії. Для цього був розроблений концепт спеціального меблевого блоку, що здатний трансформуватися під відповідні потреби власника. Управління механізмами здійснюється за допомогою жестів, за яких стежать сенсори, а продуманість конструкції вражає своїми можливостями. Незважаючи на те, що проект CityHome далекий від завершення, його розроблювачі впевнені в тому, що він виявиться надзвичайно корисний.

Система голосового управління розумним будинком

Рисунок 2.5 – Зовнішній вигляд системи голосового управління розумним будинком

На Кікстартері стартував проект пристрою, за допомогою якого можна автоматизувати роботу практично будь-якої домашньої електроніки. Новинка одержала назву Homey і створена на базі комп'ютера Raspberry Pi, що відкриває досить широкі можливості для її розвитку, однак і базовий функціонал виглядає досить вражаючим. Ключова особливість контролера – голосове управління підключеними до нього пристроями. Homey оснащений різними модулями бездротового зв'язку, такими як Wi-Fi (802.11b/g/n), Bluetooth 4.0, NFC, ZigBee, Z-Wave і інфрачервоним портом, які дозволяють об'єднати більшу частину домашньої електроніки в єдину систему. Проект зібрав практично всю необхідну для виробництва суму й Homey повинен надійти в продаж у квітні наступного року за ціною близько 400 євро.

PlantLink

Рисунок 2.6 – Зовнішній вигляд PlantLink

Цей корисний пристрій починав своє існування в якості одного із численних проектів Kickstarter. Необхідна сума була успішно зібрана в строк, і зараз воно випускається серійно. PlantLink – це бездротовий датчик вологості ґрунту, що у міру необхідності самостійно буде нагадувати про те, що настав час полити ваші кімнатні рослини. Втім, гаджет може використовуватися не тільки будинку, але й у саду.

SAM – розетка, що управляє побутовими приладами

Рисунок 2.7 – Зовнішній вигляд SAM

Ще одним свіжим проектом kickstarter у рамках концепції розумного будинку стала SAM – розетка, що може управляти живленням підключених до неї приладів. Гаджет буде випускатися у двох форм-факторах – мережного фільтра (SAM Strip) і компактного адаптера з одним слотом (SAM Plug). Обоє мають убудований набір з датчиків світла, руху, і температури, що дозволяє запрограмувати включення й вимикання споживачів на зміну навколишнього оточення. Наприклад, якщо температура підніметься вище 25 градусів увімкнеться вентилятор, а при недоліку світла – лампа. Всі налаштування здійснюються за допомогою смартфону. Мережний фільтр, крім усього іншого, має ще й два USB-порти, які можуть використовуватися для зарядки. У роздрібному продажі SAM Strip і SAM Plug будуть коштувати $109 і $49 відповідно.

Розумний термостат Tado

Рисунок 2.8 – Зовнішній вигляд Tado

Tado – це цікавий гаджет, що дозволить управляти навіть застарілими домашніми кондиціонерами за допомогою смартфону. Він має інфрачервоний порт, за рахунок якого працює як посередник. Для зв'язку із самим смартфоном є модулі Bluetooch і WiFi. Крім того, є присутнім і датчик руху, що автоматично включить кондиціонер, коли хтось зайде в приміщення. Управляти функціями пристрою можна не тільки через смартфон, але й за допомогою сенсорної панелі. Проект представлений у рамках kickstarter і збирає гроші для серійного виробництва. У роздрібному продажі термостат з'явиться в серпні за $ 150.

Microsoft Cortana для управління розумним будинком

Фахівці Onion.io продемонстрували кілька варіантів побутового застосування інтелектуального голосового помічника Cortana, що ввійшов в останню версію операційної системи Windows Phone 8.1.

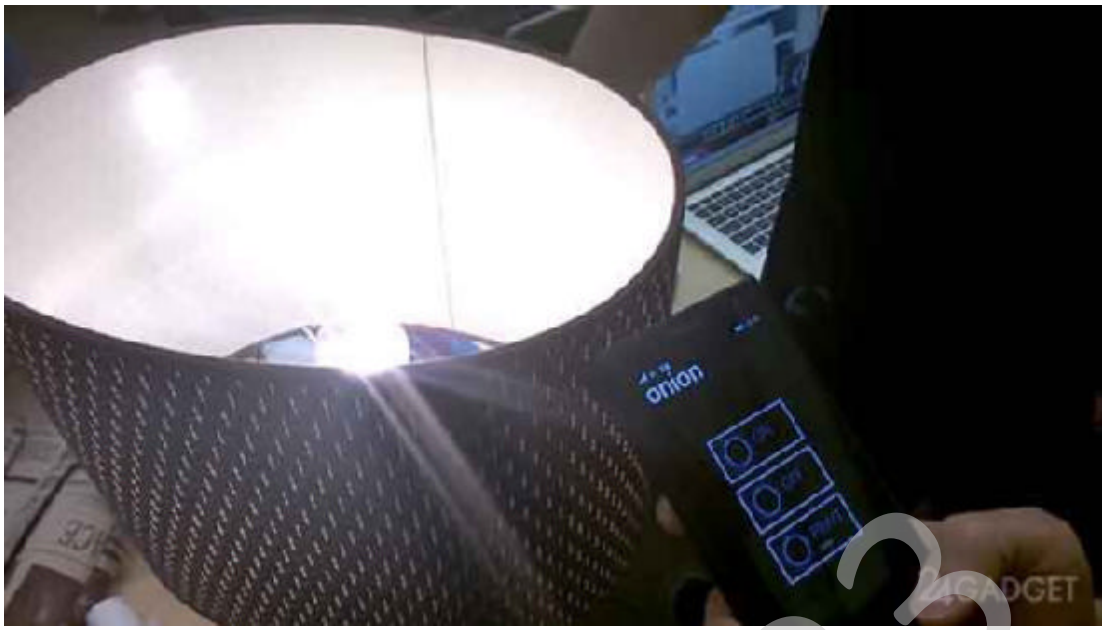

Рисунок 2.9 – Зовнішній вигляд Microsoft Cortana

Так, наприклад, програму можна подружити із розумною лампочкою Philips Hue, що дозволить управляти включенням і вимиканням світла простими голосовими командами. Також легко можна буде контролювати домашню систему опалення або кондиціювання.

Daisy.si – гаджет, що не дасть засохнути кімнатним рослинам

Якщо раніше в ледачих і забудькуватих аматорів рослин виживали тільки кактуси, то із пристроєм Daisy.si з'явиться шанс і в більше ніжних рослин. Daisy.si являє собою комбінацію з датчика вологості, датчика світла, і клапана для дозування води. Гаджет сам буде стежити за вологістю ґрунту й харчувати рослина в міру необхідності. Також це можна робити й примусово через веб-інтерфейс із будь-якої точки миру. Працює Daisy.si від пари батарейок типу AAA, яких вистачить на парі років. Конструкція має герметичний корпус, тому може використовуватися й на вулиці. Проект представлений на IndieGoGo і збирає кошти для випуску серії.

Рисунок 2.10 – Зовнішній вигляд Daisy.si

Дверний дзвінок з розпізнаванням осіб

Якщо ви шукаєте дверний дзвінок і хочете, щоб він відповідав сучасним технологіям, то зверніть свою увагу на Chui, що має функцію розпізнавання осіб. Цей пристрій може підключатися до мережі Інтернет і працювати разом з більшістю розумних електронних замків. Крім того, завдяки своїй основній особливості, а так само мікрофону й динаміку, Chui здатний передавати повідомлення конкретним людям, що зайшли до вас у гості, коли ви не хочете або не можете поспілкуватися з ними особисто. Крім того, новинкою можна управляти дистанційно за допомогою смартфону зі спеціальним додатком. Орієнтовна ціна розумного гаджету складе біля $200.

Рисунок 2.11 – Зовнішній вигляд дверного дзвінка з розпізнаванням осіб

Розумний килимок для вхідних дверей

Із чого починається розумний будинок? З розумних вхідних дверей? Ні, з розумного килимка. Так вирішив Andrew Clark і розробив пристрій, що назвав SmartMat. Його винахід складається із сенсорів тиску й контролера, що передає інформацію на смартфон хазяїна через WiFi. Завдяки точності сенсорів, власник пристрою не просто може одержати повідомлення про те, що хтось стоїть біля його дверей, але й інформацію про те, хто це: людина або вихованець, що повернувся із прогулянки. Отримані дані так само можна зберігати у пам’яті й створювати для них різні профілі, які можна привласнити конкретним гостям. Придбати SmartMat можна буде за ціною біля $100 у серпні цього року.

Рисунок 2.12 – Зовнішній вигляд розумного килимка для вхідних дверей

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об'єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об'єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на

палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз

даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

TR-069 (скорочення від Technical Report 069) – технічна специфікація, що описує протокол управління абонентським устаткуванням через глобальну мережу – CWMP (CPE WAN Management Protocol). Стандарт був опублікований в 2004 році консорціумом DSL Forum, перейменованим пізніше в Broadband Forum. Ціль – стандартизація й уніфікація принципів і підходів до управління абонентським устаткуванням різних виробників.

CWMP є протоколом прикладного рівня, що використовує як інструмент передачі інформації SOAP (Simple Object Access Protocol) – надбудову над HTTP. Всі дані передаються у форматі XML.

Відповідно до специфікації на території провайдеру повинен бути розташований сервер автоконфігурації (ACS – Auto Configuration Server), що організує взаємодію з абонентським устаткуванням, що здійснює обробку запитів від пристроїв і здатного підключати додаткові сервіси. Сесія може бути ініційована як з боку CPE, так і з боку ACS.

Для того, щоб було можливо управління пристроєм, він повинен мати IP-адресу незалежно від типу цього пристрою (Bridge, Router, IP-Phone). Для забезпечення захищеного з'єднання в TR-069 використовується SSL і TLS.

TR-069 підтримує наступні функції:

- Конфігурація. Мова йде як про початкову конфігурацію, так і автоконфігурації вже працюючого пристрою або внесенні змін у налаштування.
- Управління версіями ПЗ і його відновлення.
- Аналіз log-файлів, продуктивності й діагностика пристрою.
- Виконання збережених процедур

– Вибір CPE для обслуговування може здійснюватися по різних умовах. Наприклад, конкретний пристрій, по виробнику, моделі або версії ПЗ.

Використовувати автоконфігурацію можливо при будь-якому способі придбання пристроїв:

– Устаткування надається під час підписання договору.

– Устаткування купується абонентом у вигляді комплекту підключення до мережі оператора.

– Устаткування купується абонентом самостійно й не має попередніх налаштувань на мережу оператора.

Протокол CWMP визначає принципи взаємодії між абонентськими пристроями (CPE) і сервером Auto-Configuration Server (ACS). ACS, згідно TR-069, є мережним пристроєм і являє собою сервер додатків, що автоматизує реалізацію основних методів управління абонентськими пристроями (CPE):

– Автоматичне налаштування й динамічне реконфігурування сервісів (послуг, надаваних абонентам оператором, таких як доступ в Інтернет, VoIP, IPTV).

– Управління програмним забезпеченням (firmware) CPE.

– Моніторинг стану й параметрів продуктивності CPE.

– Діагностика.

Состав специфікації TR-069

Під формулюванням TR-069 звичайно розуміють весь комплекс специфікацій, розроблених Broadband Forum в області управління CPE. Прикладена таблиця визначає состав зазначених специфікацій. Актуальні версії відповідних документів наведені на сайті www.broadband-forum.org.

TR-046 – Auto-Config: Architecture & Framework – Визначає основні принципи автоматизованого конфігурування кінцевих пристроїв, підключених за технологією DSL.

TR-069 – CPE WAN Management Protocol v1.1 – Специфікація протоколу CWMP.

TR-098 – Internet Gateway Device Data Model for TR-069 – Описує модель даних CWMP для шлюзів доступу в Інтернет.

TR-104 – DSLHomeTM Provisioning Parameters for VoIP CPE – Описує модель даних CWMP для пристроїв VoIP.

TR-106 – Data Model Template for TR-069- Enabled Devices – Визначає об'єктну структуру й вимоги до моделі даних CWMP.

TR-111 – DSLHomeTMApplying TR-069 to Remote Management of Home Networking Devices – Визначає функції управління пристроями локальних мереж, у тому числі за NAT.

TR-135 – Enabling Network Throughput Performance Tests and Statistical Monitoring – Визначає об'єкти CWMP, що забезпечують рішення завдання моніторингу продуктивності й контролю доступності CPE.

TR-140 – TR-069 Data Model for Storage Service Enabled Devices – Описує модель даних CWMP для пристроїв зберігання.

TR-142 – Framework for TR-069 enabled PON Devices – Описує модель даних CWMP для пристроїв PON і підключених до них оптичних пристроїв.

TR-143 – Enabling Network Throughput Performance Tests and Statistical Monitoring – Визначає об'єкти CWMP, що забезпечують рішення завдання моніторингу продуктивності й контролю доступності CPE.

TR-157 – Component Objects for CWMP – Розширення об'єктної моделі CWMP відповідно до нових технологій і можливостями домашніх мереж.

TR-181 – Device Data Model for TR-069 – Визначає єдину модель даних для всіх пристроїв, що підтримують TR-069.

TR-196 – Femto Access Point Service Data Model – Описує модель даних CWMP для пристроїв femto-cell.

«Південний» інтерфейс (southbond) забезпечує реалізацію перерахованих вище функцій управління стосовно CPE. Спочатку передбачалося управління CPE з використанням технології DSL. У цей час зусиллями як самого Broadband Forum, так і інших організацій (Home Gateway Initiative, Digital Video

Broadcasting), до складу керованих по TR-069 CPE увійшли інтегровані пристрої доступу (IAD), PON і пов'язані з ними оптичні пристрої, VoIP-пристрої, приставки IPTV, інші пристрої домашніх мереж.

«Північний» (northbound) інтерфейс забезпечує взаємодію ACS з іншими системами OSS/BSS провайдеру в рамках реалізації єдиних наскрізних процесів управління послугами.

Функції управління

Ключова й одна з найбільш затребуваним бізнесом функцій управління CPE, обумовлених TR-069 – автоматичне налаштування й динамічне реконфігурування сервісів. Специфікація визначає можливість як первинного, так і повторного конфігурування CPE, наприклад, по запиті абонента або при зміні тих або інших параметрів послуги.

TR-069 визначає можливість виконання операцій конфігурування як стосовно одному конкретного CPE, так і до групи, об'єднаних одним або декількома загальними ознаками, такими як виробник CPE, модель, версія firmware і т.д.

Підтримується можливість роботи з опціональними наборами параметрів послуг (наприклад, з параметрами, що визначають платежі, функції Батьківського контролю), включення яких вимагає щодо більше високого рівня доступу, у тому числі з використанням механізму цифрового підпису. Функція управління програмним забезпеченням CPE забезпечує виконання завантаження ПЗ на пристрій. Протокол визначає механізми ідентифікації версій керованого ПЗ, ініціації (з ініціативи ACS або по запиті CPE), виконання й завершення завантаження файлів образів, логування й оповіщення служби експлуатації про результативність виконання завантаження.

Крім функцій безпосереднього управління конфігурацією CPE, протокол CWMP визначає методи надання доступу до інформації, що може бути використана сервером ACS для моніторингу статусу й продуктивності CPE.

Протокол CWMP також визначає набір механізмів, які дозволяють CPE самостійно сповіщати ACS про зміни свого стану.

Крім можливостей моніторингу CPE, CWMP надає також механізми діагностики, у тому числі состав параметрів, які можуть містити діагностичну інформацію, методи надання діагностичної інформації. Істотною відмінністю від SNMP-протоколу є можливість для ACS (виконуючого роль менеджера) не тільки запросити діагностичну інформацію з CPE, але й одержати неї в необхідному обсязі від самого CPE (у випадку SNMP можливе одержання тільки SNMP- трапа, всю іншу інформацію менеджер повинен самостійно запитувати в агента).

На додаток до перерахованим вище функцій CWMP надає механізми автоматичної автентифікації й авторизації на веб-сайтах оператора залежно від ідентифікатора й типу використовуваного для доступу CPE (докладніше відповідний механізм описаний у додатку D до специфікації TR-069).

Архітектура CWMP

Архітектура протоколу CWMP містить у собі:

- Стек протоколів CWMP, використовуваний для організації взаємодії між ACS і CPE.
- Параметри CWMP.
- Процедури CWMP.

Стек протоколів

Протокол CWMP реалізований як комплекс стандартних і спеціально розроблених протоколів. Структура стеку:

- CPE/ACS Management Applicatioin.
- RPC.
- SOAP 1.1.
- HTTP.
- SSL/TLS.
- TCP/IP.

Параметри CWMP

Параметри CWMP являють собою модель даних, структура якої визначена іншим документом Broadband Forum – «TR-106: Data Model Template for TR-069-Enabled Device». Основне призначення параметрів – надання даних ACS про характеристики й стан CPE, управління їхньою конфігурацією. Параметри можуть бути визначені як read-only або read-write. Параметри read-only можуть використовуватися сервером ACS для визначення специфічних характеристик CPE, поточного стану CPE або одержання накопиченої статистики. Параметри read-write дозволяють серверу ACS змінювати конфігурацію CPE. В CWMP всі параметри об'єднані в ієрархічну структуру. Ця структура даних в CWMP представлена у вигляді об'єкта. Кожний об'єкт містить один параметр або набір параметрів. Кожний CPE має тільки один головний об'єкт (root), залежно від типу пристрою приймаюче значення Device або InternetGatewayDevice.

У більшості випадків, головний об'єкт містить у собі три елементи:

- Common Objects.
- Components.
- Service Objects.

Об'єкт Components містить параметри, що забезпечують різні функції TR-069. Їхня специфікація наведена в окремих документах BroadBand Forum.

CPE/ACS Management Application – додаток, що запускається на CPE або ACS, що реалізує функції CWMP. Додаток не специфікується CWMP і може являти собою сервіси, GUI додатка й т.п. RPC – набір методів RPC, використовуваних при взаємодії між ACS і CPE, описаний у специфікації CWMP.

SOAP – всі повідомлення, передані між ACS і CPE, конвертуються у формат XML. Використання SOAP дозволяє забезпечити платформонезалежність рішення, піти від специфіки реалізації конкретних додатків.

HTTP – обраний як транспортний протокол для запитів SOAP за його поширеність. Передбачається, що в більшості випадків налаштування міжмережних екранів припускають пропуск трафіку по портах http, відповідно,

при впровадженні CWMP не буде потрібно істотного перегляду корпоративних політик інформаційної безпеки SSL/TLS – при передачі даних між CPE і ACS використовуються відповідні методи шифрування трафіку, що забезпечують конфіденційність і цілісність переданих даних. Використовується автентифікація взаємодіючих по протоколі CWMP сторін з використанням сертифікатів.

TCP/IP – всі повідомлення, передані між ACS і CPE, відповідно до TR-069 повинні віддаватися з використанням протоколу TCP для забезпечення їхньої гарантованої доставки.

Використання TCP також визначається необхідністю роботи в умовах використання NAT.

ServiceObjects містить об'єкти для кожного типу послуг, забезпечуваних конкретною моделлю CPE. Відповідно, для мультисервісних CPE визначається кілька об'єктів відповідного типу. Для основних типів послуг параметри формалізовані й оформлені у вигляді відповідних рекомендацій BroadBand Forum.

Для спрощення операцій масового управління CPE, відповідно до TR-069, для ACS визначається поняття профілю. Під профілем розуміється набір вимог, яким повинні задовольняти значення параметрів одного або декількох CPE. Параметри конфігурації сервісу, його підключення або відключення можуть бути зібрані в єдиний профіль. Кожний CPE може мати більше одного профілю – залежно від кількості й типу надаваних абонентові сервісів. Звичайно для кожного типу CPE виділяється один базовий профіль, що визначає основні параметри його роботи, а також трохи додаткових, отриманих на основі базового й специфічні для типів, що включають у себе, сервісів параметри.

CommonObjects містить параметри, які визначають тип CPE. Параметри вітки CommonObject використовуються для ідентифікації CPE на ACS і містять у собі:

- DeviceInfo.
- ManagementServer.

– GatewayInfo.

– Time.

– Config.

– UserInterface.

– LAN.

Завершуючи короткий огляд параметрів, необхідно відзначити, що CWMP не регулює число параметрів, підтримуваних конкретною реалізацією CPE. Кожний виробник може додати свої, специфічні параметри, що щонайкраще реалізують функції управління конкретним типом і моделлю устаткування. Протокол CWMP у свою чергу визначає лише набір основних параметрів, що дозволяють реалізувати функції уніфікованого управління гетерогенною мережею оператора.

3.2 Розробка структурної схеми

Домашні мережі стають усе більше масштабними й складними. Вони зв'язують різноманітні взаємодіючі між собою пристрою: ігрові консолі, телеприставки, телефони VoIP і ПК. Зважаючи на те, що щодня в системі безпеки виявляється безліч уразливостей, наявність яких може вплинути на потоки переданого в домашній мережі трафіку, цими пристроями необхідно управляти, щоб забезпечити належна якість сервісу й підвищити надійність мережі. Це завдання спрощується за рахунок надання за допомогою TR-069 єдиної платформи для ефективного управління всіма мережними пристроями.

Зростаюче сімейство розширюваних і керованих модульних «моделей даних» TR-069 визначає широкий спектр функцій пристроїв і програмних модулів. Ці моделі, по-перше, указують, що пристрій ставиться до категорії підключених і повинне розпізнаватися мережею, а по-друге, надають відомості про тип пристрою – VoIP, телеприставка, шлюз або фемтостільникова точка доступу.

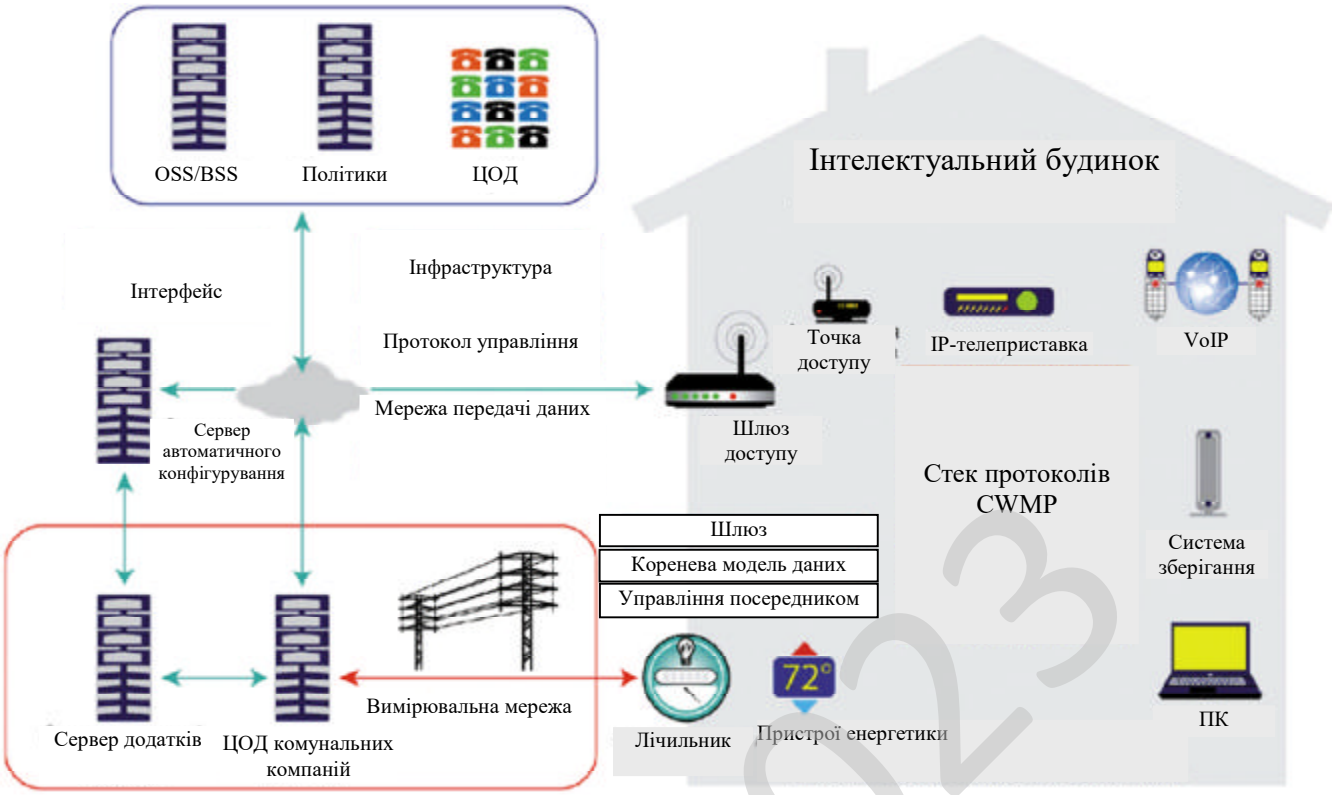

Рисунок 3.1 – Структурна схема системи

Своя модель даних передбачена й для управління посередниками – завдяки їй, мережа може підтримувати такі об'єкти, як лічильники електроенергії, пристрої для забезпечення безпеки й домашнього моніторингу. Список пристроїв продовжує поповнюватися, і TR-069 регулярно обновляється для підтримки всі нових їхніх типів. Його ефективність не залежить від локального методу доступу (DSL, кабельна, бездротова мережа й т.д.), він може працювати з будь-якими з'єднаннями й протоколами в домашній мережі (G.hn, HomePlug, OAM, і ін.).

Виводячи на ринок новий мережний пристрій, потрібно навчити його «розмовляти» на одній мові з мережею, щоб остання могла ідентифікувати новий пристрій, що підключається, підтвердити його тип і забезпечити швидкість і стабільність роботи, необхідні для виконуваних на ньому додатків. Відповідність пристрою стандарту TR-069 і вимогам моделі даних гарантує ефективне виділення ресурсів і одержання замовником підтримуваних можливостей.

Мова йде не тільки про управління шлюзом широкополосного доступу, але й про забезпечення взаємодії з іншими пристроями, такими як телеприставки, IP-телефони й мережні системи зберігання (NAS). Важливі переваги одержують і провайдери: вони можуть віддалено набудовувати конфігурацію, управляти різними новими пристроями, що підключаються, і здійснювати діагностику. А для клієнтів це означає, що вони не будуть зіштовхуватися з якими-небудь проблемами при виділенні ресурсів, усуненні несправностей і модернізації устаткування.

Одержання клієнтським устаткуванням (CPE) сертифікації Broadband Forum BBF.069 ясно й недвозначно говорить про те, що галузь підтверджує його готовність до виходу на ринок. Тепер постачальники устаткування й систем зможуть знизити витрати на тестування, спростити просування продуктів і скористатися новими можливостями, що дозволяють значно розширити охват ринку.

Дана програма націлена на рішення наступних завдань:

– перевірка відповідності пристроїв протоколу TR-069 і їхньої готовності до виходу на ринок;

– спрощення інтеграції домашніх мережних систем і устаткування з мережами провайдерів;

– поліпшення керованості пристроїв, підвищення їхньої економічності й простоти управління для провайдерів, яким доводиться мати справа з більшим числом пристроїв, що підключаються до мережі.

Програма сертифікації Broadband Forum дає істотні вигоди всім учасникам ланцюжка поставок. Введення сертифікації по TR-069 полегшує операторам і сервісам-провайдерам прийняття інвестиційних рішень. Ця програма гарантує сумісність із мережею і якість, а оператори й сервіспровайдери можуть бути впевнені в тому, що сертифіковані пристрої CPE будуть краще взаємодіяти з їхніми системами віддаленого управління й конфігурування (ACS).

Виконаний ABI Research аналіз дозволяє припустити, що до 2016 року обсяг поставок базового устаткування для домашніх мереж (шлюзів/маршрутизаторів, мостів, мережних плат (NIC), що вбудовуються мережних адаптерів (LAN) і мережних систем зберігання (NAS)), а мультимедійних пристроїв, що підключаються також до мережі, (клієнтських пристроїв з мережними інтерфейсами, включаючи комп'ютери й мобільні пристрої) перевищить мільярд одиниць. По оцінках IHS iSuppli, в 2014 році в усьому світі число проданих шлюзів і тонких клієнтів досягло 4,2 млн штук, у той час як роком раніше цей показник становив 345 тис. Згідно із прогнозами, у наступні два роки обсяг продажів буде стрімко збільшуватися – до 6,7 млн штук в 2015 році, до 10,4 млн штук в 2016-м і до 12,6 млн штук до 2017 року.

Виробники побутової електроніки, клієнтського устаткування, «розумних» лічильників і пристроїв можуть продавати більше продуктів, якщо вони сертифіковані на відповідність TR-069, оскільки це гарантує, що устаткування легко інтегрується в домашню мережу і їм можна ефективно управляти віддалено.

Контент-провайдери одержують аналогічну можливість для збільшення продажів – споживання «мультиекраного» контенту (для екранів різного типу) росте з розвитком екосистеми пристроїв.

Сервіси-провайдери можуть продавати більше керованих пристроїв і послуг, дістаючи прибуток не тільки від базового сервісу широкополосного доступу. А завдяки забезпечуваному TR-069 віддаленому управлінню знижуються витрати на підтримку мережі й устаткування, підвищується лояльність клієнтів.

Комунальні компанії можуть розширити спектр своїх послуг, запропонувавши на основі TR-069 функції управління будинком і моніторингу.

Тим часом, завдяки різним ініціативам, використання стандарту TR-069 для побудови в будинках мереж «розумних» лічильників поліпшить управління енергоресурсами й потенційно знизить, що виставляються в рахунках суми.

Таким чином, TR-069 надає провайдерам можливості моніторингу, управління й контролю за зростаючим числом усе більше складних підключених пристроїв, терміналів і устаткування. Він дозволяє сервіспровайдерам підтримувати високий рівень якості, пропонувати привабливі для користувачів можливості, надає їм інструменти для поліпшення обслуговування клієнтів і створює передумови для успішної комерціалізації мультиекраних систем і комбінованих послуг.

За останні роки Україна стала одним із ринків широкополосного доступу, які швидко розвиваються, уступаючи лише Китаєві й Індії. Так, наприклад, в 2014 році його ріст склав 28,3% по числу абонентів, а широкополосний доступ охоплює приблизно 15% населення. Більшість користується DSL, але одночасно збільшується кількість підключень по оптичному волокну й через кабельні модеми.

Користувачі широкополосних мереж всі частіше хочуть одержувати мультисервісні можливості й підключають до таких мереж безліч нових пристроїв. У цей час росте популярність додатків IPTV, і хоча поки в Україні ними користуються менш 2% населення, за останні чотири роки число передплатників IPTV різко збільшилося й тепер наближається до 3 млн чоловік.

Для сервісів-провайдерів, що працюють в Україні, критично важливим є впровадження світових стандартів мережної архітектури й управління, а TR-069 з різноманітними моделями даних гарантує експонентний ріст українського ринку й ефективне обслуговування населення.

З появою сертифікації Broadband Forum BBF.069 Certification стандарт TR-069 стає ключовим інструментом для подальшого розширення потенціалу й збільшення кількості підключених домашніх пристроїв в усьому світі й особливо в Україні. Сервіси-провайдери можуть внести свою лепту в цей процес, дотримуючись стандарту TR-069 у своїх платформах управління мережами й вимагаючи в запитах RFP сертифікації по BBF.069 CPE.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна структура розробленої мережі складається з наступних блоків, як взаємодіють з блоком функцій CWMP:

– блок функцій, які відповідають за ручне управління інтелектуальним домом, тобто дозволяють через розроблене програмне забезпечення за допомогою пультів, або персонального комп'ютера управляти усіма болками інтелектуального дому;

– блок функцій, які відповідають за роботу з мультимедіа, тобто дозволяють по локальній мережі всередині будинку передавати дані на різного виду кінцеві пристрої (плазмений телевізор, сенсорні панелі й т.і.);

– блок функцій, які відповідають за безпеку, що дозволяє в автоматичному режимі знімати дані з датчиків і по мережі передавати їх до охоронного сервера, що приймає рішення, яку функцію виконати ;

– блок функцій, які відповідають за автоматизацію, що дозволяють незалежно від людини виконувати роботи по підтримці дому у належному стані, тобто сервер автоматизації приймає рішення включати той або інший прибор.

Блок функцій, які відповідають за ручне управління інтелектуальним домом включає в себе наступні функціональні блоки:

– управління з екрану ЕОМ;

– релейне управління;

– діммування;

– «проходні» схеми вимикання;

– бездротове управління електрикою;

– для аудіо/відео та світла один пульт ДУ;

– віддалене управління за телефоном.

Рисунок 3.2 – Функціональна схема системи

Блок функцій, які відповідають за роботу з мультимедіа, включає в себе наступні функціональні блоки:

– бездротове аудіо/відео;

– універсальні навчаємі пульти ІЧ ДУ;

– радіотрансляція ІЧ ДУ;

– бездротове аудіо/відео спостереження;

– радіоуправління ПЗ ЕОМ.

Блок функцій, які відповідають за безпеку, включає в себе наступні функціональні блоки:

– звукова тривожна сигналізація;

– охоронна сигналізація;

– світлова тривожна сигналізація;

– тривожне оповіщення по телефону;

– світлова імітація наявності людей;

– функція «Паніка».

Блок функцій, які відповідають за автоматизацію, включає в себе наступні функціональні блоки:

– макроси та сценарії;

– за температурою повітря;

– по руху людини;

– за рівнем освітленості;

– по сухому контакту;

– за часом доби.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Ввімкнення системи CWMP.

– Читання показів датчиків.

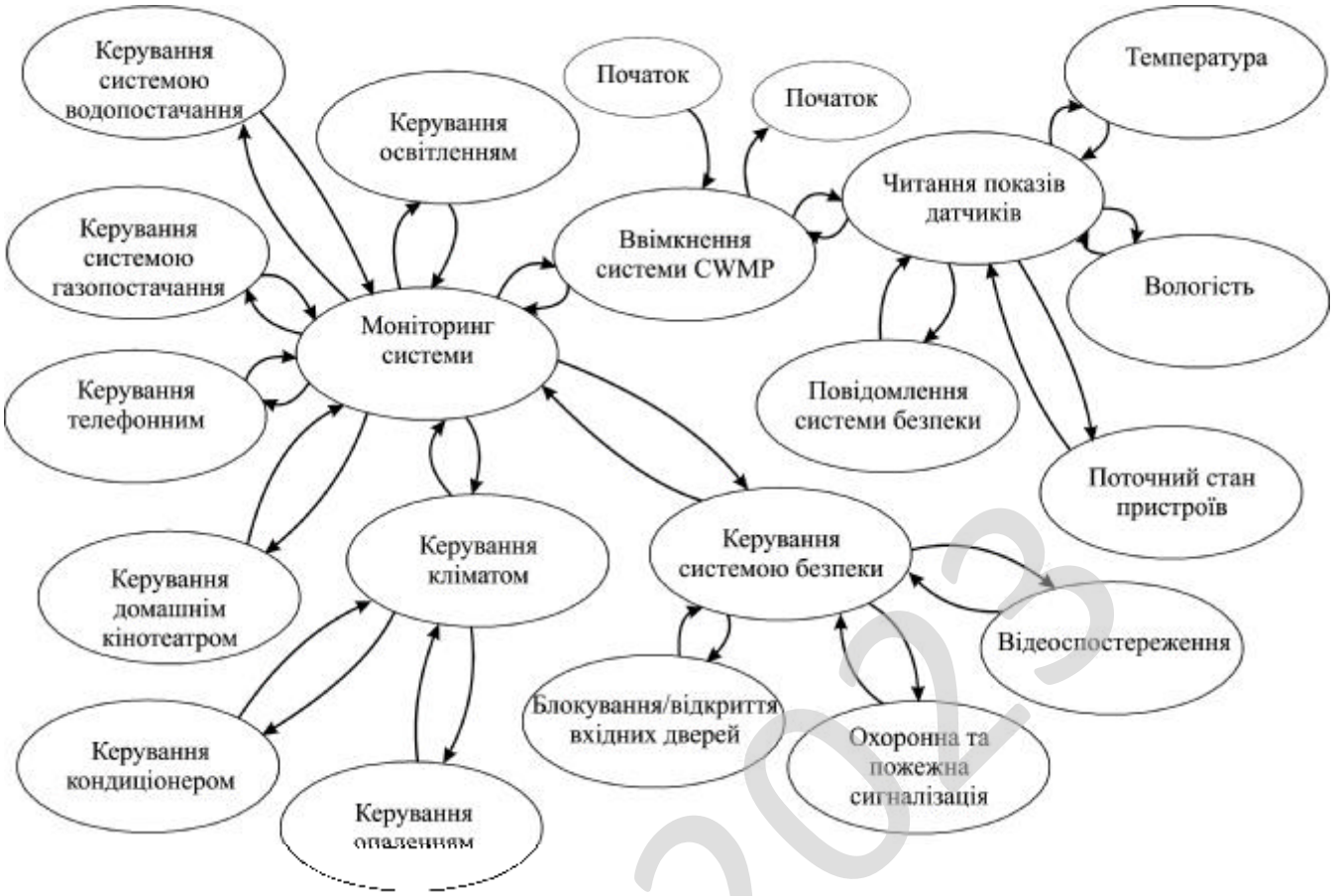

Рисунок 3.3 – Діаграма взаємодії процесів

– Температура.

– Вологість.

– Поточний стан пристроїв.

– Повідомлення системи безпеки.

– Моніторинг системи.

– Керування освітленням.

– Керування системою водопостачання.

– Керування системою газопостачання.

– Керування телефонним зв’язком.

– Керування домашнім кінотеатром.

– Керування кліматом.

– Керування кондиціонером.

– Керування опаленням.

– Керування системою безпеки.

– Блокування/відкриття вхідних дверей.

– Охоронна та пожежна сигналізація.

– Відеоспостереження.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

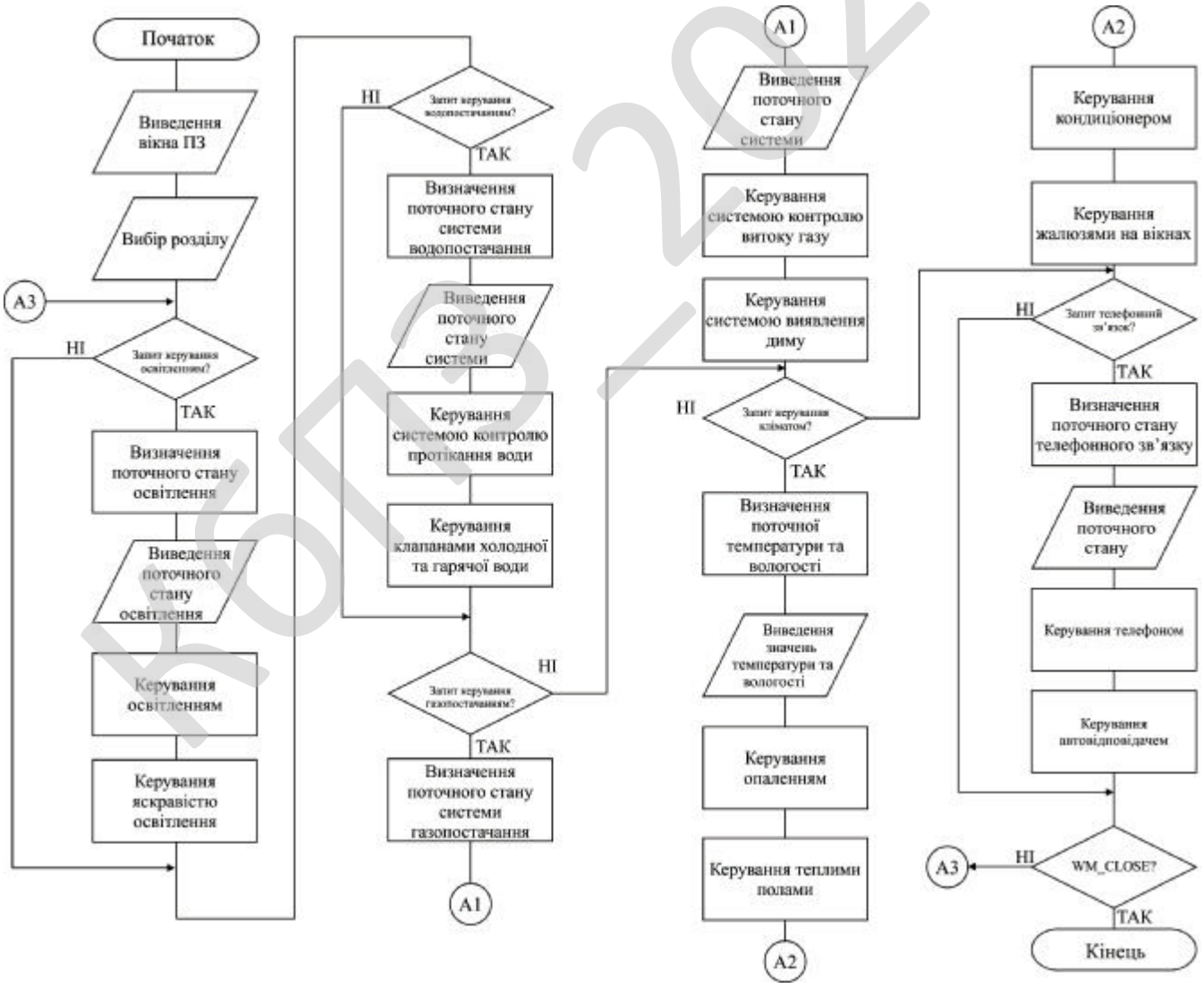

Рисунок 4.1 – Блок схема основної програми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

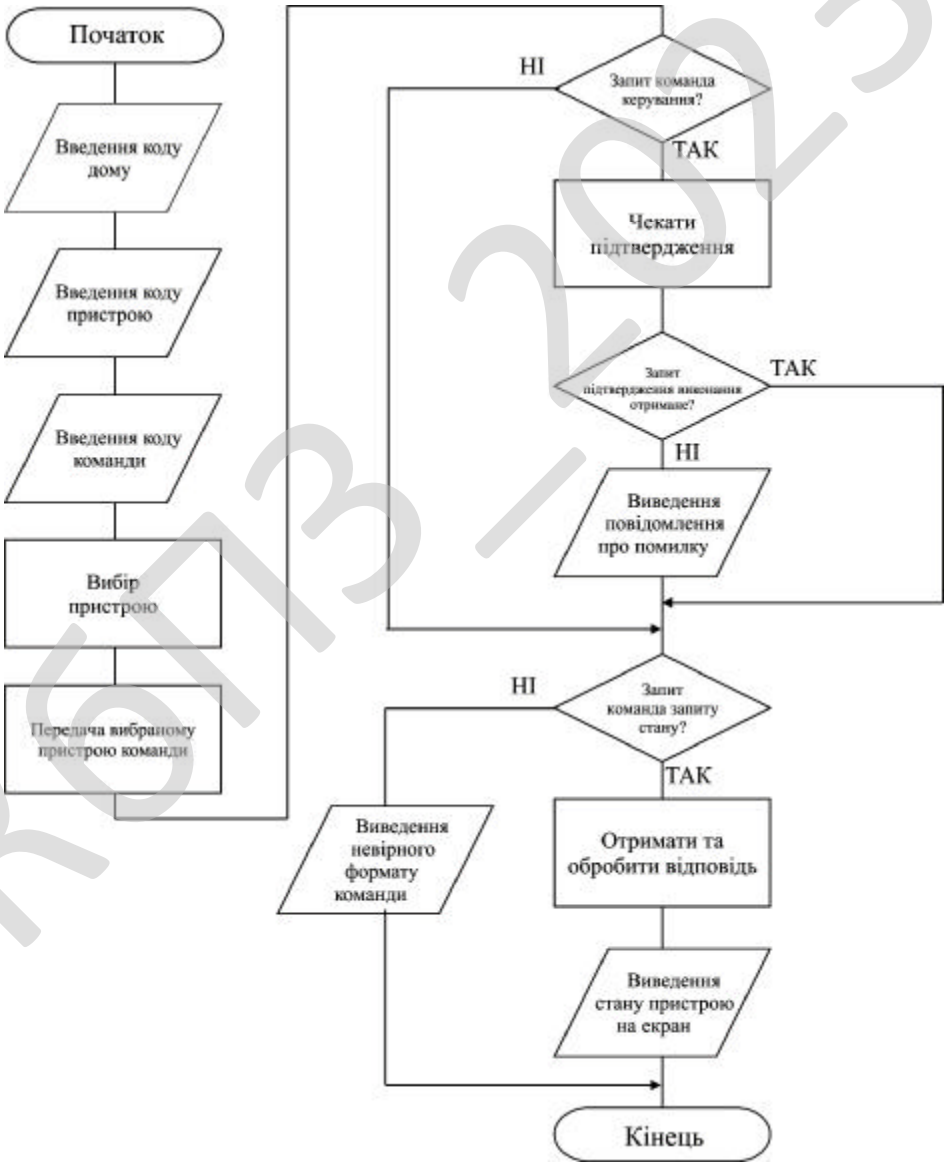

Рисунок 4.2 – Блок схема підпрограми

Опис алгоритмів функціонування системи

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом.

Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу

чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Опис використання UML. Це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Опис сигналів управління CWMP

Хоча сигнали управління CWMP передаються й приймаються безпосередньо по електричній мережі, для під'єднання пристроїв до комп'ютера можна використовувати різні топології, такі як наприклад Ethernet, RS-232, RS-485 та інші. Для вирішення поставленого у дипломному проекті завдання, було обрано інтерфейс Ethernet.

Стандарт CWMP визначає метод і протокол передачі керуючих сигналів-команд (включити, виключити, яскравіше, темніше й т.д.) по силовій електропроводці на електронні модулі, до яких підключені керовані електропобутові й освітлювальні прилади. Усього в мережу CWMP може бути об'єднане до 256 груп пристроїв з різними адресами.

З погляду логіки організації внутрімережної взаємодії всі пристрої CWMP можна розбити на дві більші групи: контролери й виконавчі модулі.

Контролери відповідають за генерацію команд CWMP і, крім ручного кнопкового керування, можуть мати убудований таймер або спеціалізований пристрій уведення зовнішнього впливу (датчик освітленості, фотоприймач інфрачервоного випромінювання від пульта дистанційного керування й т.д.).

Виконавчий модуль, виконуючи команди, передані тим або іншому контролеру, управляє комутацією електроживлення побутового або освітлювального приладу, відіграючи роль "розумного" вимикача. Найпоширеніші модулі двох типів: лампові (lamp module) і приладові (appliance module).

Конструктивно лампові модулі являють собою тиристорні регулятори потужності й забезпечують, крім функцій включення й вимикання, плавну регулювання яскравості світіння електроламп (функція діммера, від англійського слова dimmer – "реостат", "затемнювач").

Приладові модулі оснащені електромагнітними реле для перемикання живлення й не призначені для плавного регулювання подаваної на навантаження потужності.

Для передачі команди CWMP потрібно одинадцять циклів (періодів) силової напруги. Перші два цикли передають стартовий код, наступні чотири цикли представляють код будинку (з A по P) і останні п'яти циклів передають код приладу (з 1 по 16) або код функції (ВМК, ВИМК і т.д.), тобто ключовий код. Цей повний код (стартовий код + код будинку + ключовий код) завжди передається двічі безперервним блоком. Між блоками різних команд завжди повинен бути перерва в три цикли силової напруги. Виключенням із цього правила є блоки команд ЯСКРАВІШЕ/ТЕМНІШЕ, які передаються послідовно (мінімум два блоки) без затримок .

Усередині кожного блоку, код будинку й ключовий код повинні передаватися з кодами, що доповнюють до одиниці, у суміжних напівперіодах силової напруги. Наприклад, якщо одиничний імпульс переданий у першій половині періоду, то в другий не повинне бути ніякого сигналу (нульовий біт).

Нижче наведені можливі значення коду будинку й ключового коду і їхні двійкові подання. Стартовий код – це унікальний код, завжди рівний 1110 і не має суміжних напівперіодах доповнюють, що біт в, тобто значущі біти передаються на кожний перехід силової напруги через нуль.

1. HAIL запит (запит-вітання) передається для знаходження передавачів у зоні покриття. Це дозволяє виставити різні коди будинків у випадку одержання відповіді Hail Acknowledge.

2. У коді функції Pre-Set Dim, біт D8 разом із чотирма бітами коду будинку становить блок з 5 біт {D8H8H4H2H1}, що визначає абсолютний рівень діммеру.

3. Функція Extended Data (додаткові дані) передує послідовності байт (8 біт) довільної довжини, які представляють аналогові дані після аналогово-цифрового перетворення. Код функції й байти даних передаються безупинно, без пауз. Перший байт даних може вказувати на кількість байт у послідовності. Якщо при передачі в послідовності байт допущені паузи, то модуль – приймач може виконати помилкову операцію.

Функція Extended Code еквівалентна Extended Data: послідовність байт (без

пауз), які представляють додаткові коди. Це дозволяє розроблювачам використовувати більше 256 наявних кодів.

Перші 16 із ключових кодів визначають номер модуля, що надалі буде приймати й виконувати команди (ВМК, ВИМК, ЯСКРАВІШЕ, ТЕМНІШЕ) до перевизначення керованого модуля. Біт D16 називається "функціональним бітом", якщо він дорівнює 1, то передається функція, інакше код модуля.

Приведемо приклад.

Щоб включити 5-ий модуль в "будинку К", потрібно послати по електромережі наступний рядок біт:

1110010110100101011001111001011010010101100100000011100101101001
0110011011100101101001011001 10.

Ця посилка містить 94 біта, і займе 47 циклів силової напруги або 0,94 с (майже секунда!). Тому, коли ви натискаєте на кнопку ВМК, світло включається із запізнюванням. Реакція на команди "Все світло ВМК" або "Всі модулі ВИМК" помітно швидше, тому що не передається код модуля.

Опис сигналів CWMP

Технологія CWMP використовує цифрове подання сигналів керування. Інформація кодується двійковим кодом і передається по електричній мережі за допомогою високочастотних імпульсів. Кожний переданий імпульс відповідає одному біту інформації сj значенням “1”. Передача чергового імпульсу відбувається в момент часу, коли сіткова напруга приймає нульове значення.

Стандартна команда CWMP передається протягом, приблизно, 50 періодів мережної напруги частоти 50Гц або 1 сек.

Більшість переданих по мережі CWMP повідомлень містить, принаймні, два інформаційні поля: адреса пристрою, якому ця команда адресована, і властиво команду.

Підключені до електромережі пристрої CWMP приймають передані повідомлення, декодують поле адреси й, якщо він збігається з їх власним, виконують команду.

Опис команд CWMP

У протоколі CWMP передбачено шість базових команд:

1. Включити (On).
2. Виключити (Off).
3. Яскравіше (Bright).
4. Темніше (Dim).
5. Включити все світло (ALL Ligts ON).
6. Виключити всі (ALL Units OFF).

Опис інтеграції CWMP

Основним засобом інтеграції мережі CWMP із зовнішнім устаткуванням є PLC інтерфейс CWMP. Будь-який контролер, що підтримує відкритий протокол обміну з CWMP, може відправляти команди CWMP в електромережу й, навпаки, одержувати з мережі інформацію про стан пристроїв CWMP.

Опишемо вихідний код. Який відображає підключення основних функцій у програмі.

```
//відкриття вікна "Система клімат-контролю"
void __fastcall TForm_main::Button2Click(TObject *Sender)
{
Form_climate->Show();
}
//---------------------------------------------------------
//відкриття вікна "Система газопостачання"
void __fastcall TForm_main::Button5Click(TObject *Sender)
{
Form_gas->Show();
}
//---------------------------------------------------------
//відкриття вікна "Система безпеки"
void __fastcall TForm_main::Button3Click(TObject *Sender)
{
Form_security->Show();
}
//---------------------------------------------------------
//відкриття вікна "Домашній кінотеатр"
void __fastcall TForm_main::Button7Click(TObject *Sender)
{
```

```
Form_tv->Show();
}
//---------------------------------------------------------
//відкриття вікна "Телефонній зв'язок"
void __fastcall TForm_main::Button6Click(TObject *Sender)
{
Form_phone->Show();
}
//---------------------------------------------------------
//відкриття вікна "Управляння освітленням"
void __fastcall TForm_main::Button1Click(TObject *Sender)
{
Form_light->Show();
}
//---------------------------------------------------------
//відкриття вікна "Про програму..."
void __fastcall TForm_main::Button8Click(TObject *Sender)
{
Form_about->Show();
}
//---------------------------------------------------------
//відкриття вікна "Система водопостачання"
void __fastcall TForm_main::Button4Click(TObject *Sender)
{
Form_water->Show();
}
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку видно, що за допомогою головного вікна користувач здійснює вибір розділу програми, в який він хоче перейти, а саме:

1. Світло.
2. Водопостачання.
3. Газопостачання.
4. Безпека.
5. Клімат.
6. Телефонний зв'язок
7. Домашній кінотеатр.
8. Довідка.

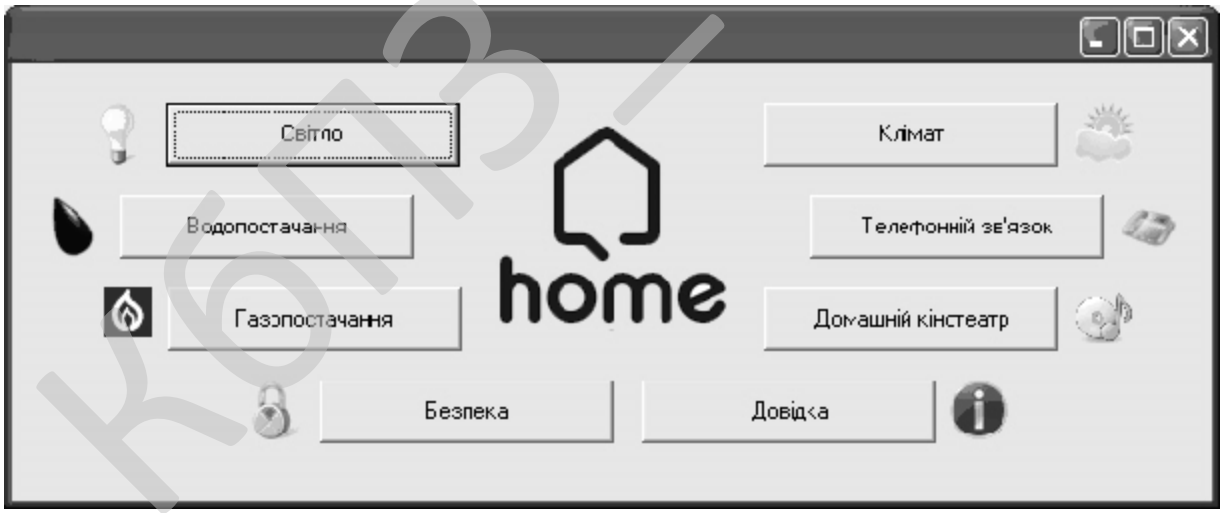

Рисунок 5.1 – Головне вікно програми

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

На рисунку 5.2 зображено вікно управління мережними пристроями інтелектуального будинку за технологією CWMP. На рисунку 5.3 зображено авторські дані розробленого програмного забезпечення.

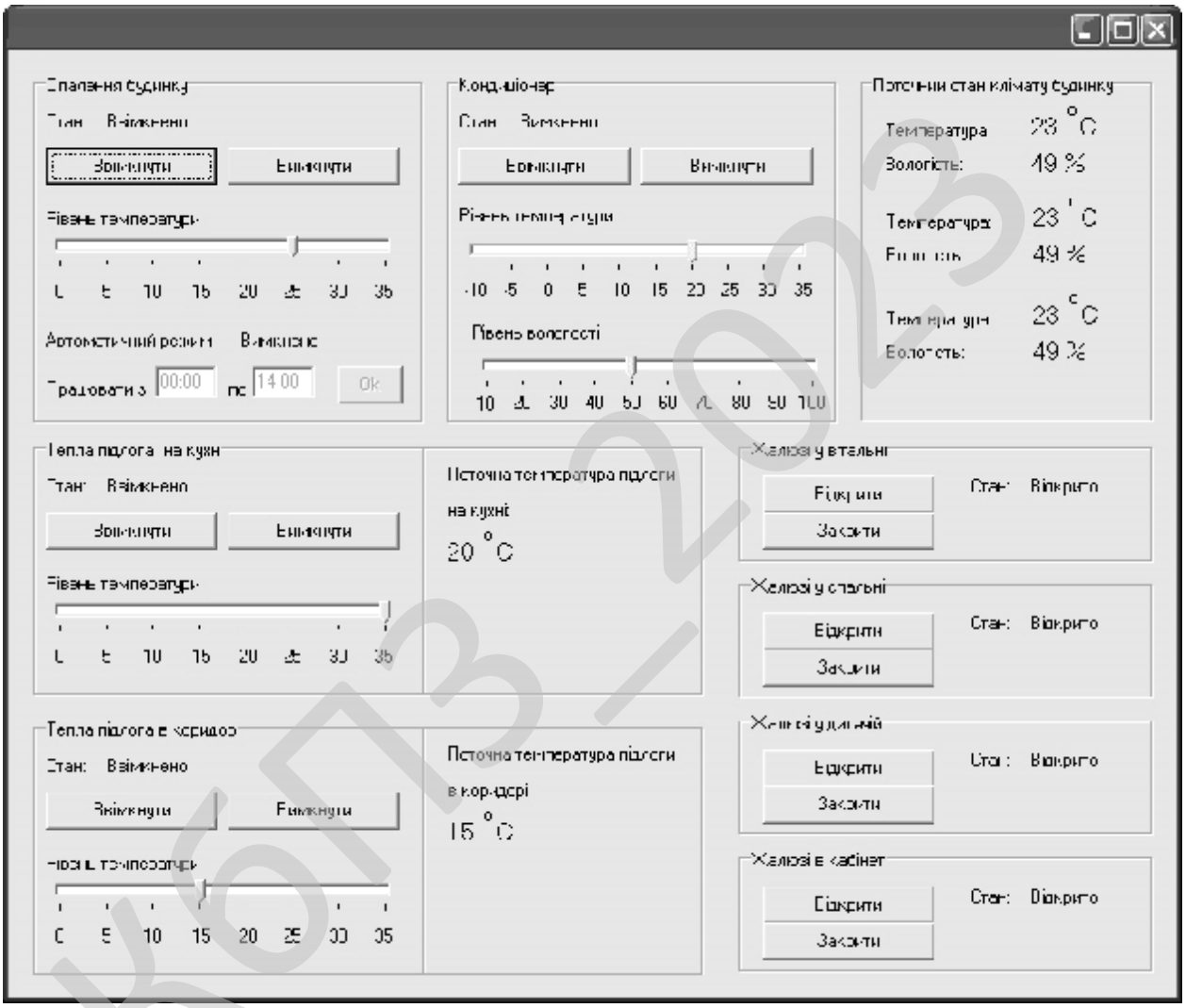

Рисунок 5.2 – Керування кліматом

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

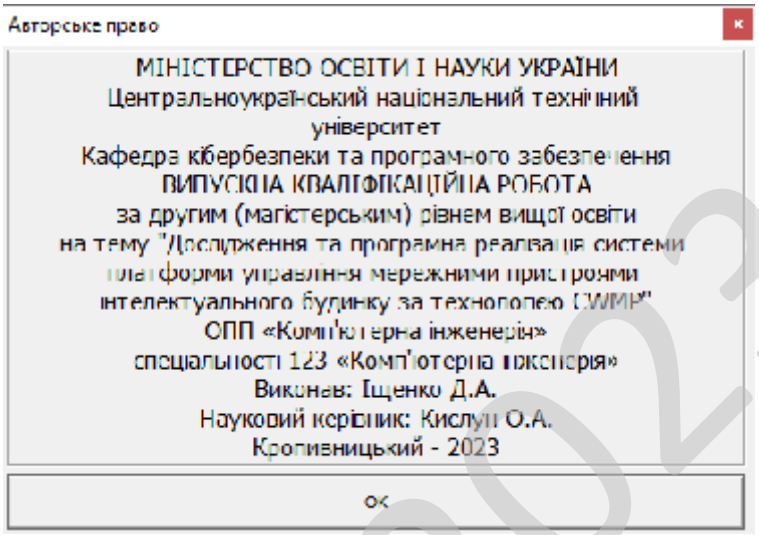

Рисунок 5.3 – Авторське право

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Метою розробки є дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Об'єктом дослідження є процес платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Предметом дослідження є методи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– Розроблено вітчизняний продукт платформи управління мережними пристроями інтелектуального будинку за технологією CWMP, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	320
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	32000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$
$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 38 = 71 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	71	Ф 7.1- 7.4
Впровадження	15	Д13
Всього	120	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{120 \cdot 1}{24\text{-}3} = 5,75 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м. п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	56,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2}, \qquad (7.6)$$

$$Ф_{др}^{с} = \frac{56 \cdot 1}{1,2} = 47 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 47/(24 \cdot 8) = 0{,}25 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	0,25	26000	6500
Продакт-менеджер	0,5	19000	9500
Інженер-програміст	5,75	23000	132250
Інженер-електронщик	0,25	21538	5384,5
Інженер-системотехнік	0,25	21538	5384,5
Адміністратор мережі	0,5	19000	9500
Системний програміст	0,25	21538	5384,5
Дизайнер WEB	0,5	19000	9500
Інженер-верстальник	0,25	21538	5384,5
Бухгалтер-економіст	0,5	19000	9500
Всього за період розробки	$R_{сп} = 9$	-	$Ф_{роб} = 198288$

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{198288}{9 \cdot 24} = 918 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу supercomp за 28.10.23 – джерело https://supercomp.kiev.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11457
Системний блок		7509
Процесор	AMD Ryzen 3 4100 (100-100000510BOX) AM4, 4 ядра, 8 потоків, 3.8 GHz, 4.0 GHz, TDP – 65 Вт, 7nm	-
Системна плата	ASRock A520M-HDV сокет – AM4, DDR4, 64 ГБ, 4733 MHz, LAN – 1 Гбіт/с, D-Sub (VGA), DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	GeForce GT710 2048Mb Afox PCI-Express 2.0, 2 ГБ, GDDR3, 64 Bit	-
Жорсткий диск	SSD 2.5" 240GB Mibrand (MI2.5SSD/SP240GB) Серія – Spider, 240 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot	-
Корпус	Vinga CS104B-500W, Miditower, ATX, Micro – ATX, Mini – ITX, PSU – 500 Вт	-
Кулер	Vinga CL3011	–
Кардрідер внутрішній	-	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V (BE525-RS)	1348

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	1000
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	45500	25	11375
Всього по групі	146999	-	31875
7. Нематеріальні активи	32000	10	3200
Разом	$К_р$ = 2652653		$А_р$ = 242302

Примітка: вартість автомобіля взята по даним з прайс-листа автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 918 \cdot 120/320 = 342 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 342 \cdot 10 \cdot 0{,}01 = 34 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(342+34) = 83 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 32,15 грн./шт.

$$З_{М2} = 32{,}15 \cdot 100 = 3215 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (40 + 3215 + 1702)/320 = 15 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 320 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 242302 \cdot 1/(320 \cdot 12) = 63 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 342+34+83+51+15+51+63 = 639 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 639 = 320 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	342
2. Додаткова зарплата виконавців	$З_д$	34
3. Відрахування на соціальні потреби	$С_{оц}$	83
4. Загальногосподарські витрати	$Г_{осп}$	51
5. Витрати на матеріали	$З_М$	15
6. Освоєння нових операційних систем, мов програмування	$О_п$	51

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	63
8. Повна собівартість програмного забезпечення	$С_п$	639
9. Плановий прибуток	$П_р$	320
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	959
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	191,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1150,8

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1151
Всього капітальних витрат	–	1151

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	12078	6710
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	576
Всього витрат за рік	$І$	12078	7286

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 90 годин на рік до 50 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 90 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 12078\ грн,$$

до:

$$З_{р\,нов} = 50 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 6710\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = З_{ел\ нов}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	1151	–	575,5
Всього відрахувань	-	–	1151	–	575,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot K_{р_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (959 - 639) \cdot 320 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 +$$
$$+0,25 \cdot 49499 + 0,2 \cdot 97500 + 0,1 \cdot 32000) \cdot 1/12 = 82208\ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника.

$$T_в = \frac{2652653}{(959-639)\cdot 320\cdot 12/1} = 2{,}16 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	320
2. Повна собівартість розробленої програми	Грн.	639
3. Ціна розробленої програми	Грн.	959
4. Плановий прибуток від реалізації розробленої програми	Грн.	320
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2652653
7. Загальний прибуток від реалізації програмної продукції	Грн.	102400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	2,16
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1151
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4217
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,24

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (12078\text{-}7286)\text{-}0{,}5 \cdot 1151 = 4217 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{1151}{(12078 - 7286)} = 0{,}24 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [4], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3,5
Довжина	3,5
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,1
Обсяг, V	$м^3$	не менше 20.0	18,4

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працює 2 людей. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про

затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», а об'єм – ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	45-60	0,12
Тепла	23-25	50-70	0,1	23-25	52-70	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP Laser 107a, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути

освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 3,5 м.; довжина –3,5 м.; висота – 3 м.

Розрахунок штучного освітлення проводемо за методом коефіцієнта використання світлового потоку.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою *[9]*:

$$F=ESKZ/n,$$

де: *F* – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення (у нашому випадку *S*=3,5 х 3,5 = 11,9 $м^2$);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін.}$)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють $\rho_{стін}$ = **50% і $\rho_{стелі}$ = 50%** *[6]***.**

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: S – площа приміщення, $S = 11{,}9\ м^2$;

h – розрахункова висота підвісу, $h = 3\ м$;

A – ширина приміщення, $A = 3{,}5\ м$;

B – довжина приміщення, $B = 3{,}5\ м$.

Підставимо всі значення у формулу та визначемо індекса приміщення: $i=0{,}57$.

Знаючи індекс приміщення, за знаходимо $n = 0.29$ *(з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу [6])*. Підставимо всі значення у формулу, визначемо світловий потік: $F=13182$ *Лм.*

Для штучного освітлення приміщення використовуються *LED панель MAXUS ASSISTANCE PRO 80W 5000K WHITE (M1052480531)*, світловий потік яких $F_л = 8000$ Лм.

Число світильників визначається по формулі **[5]**:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 13182/ 8000=1{,}6\ шт.$$

Приймаємо необхідну кількість світильників 2 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного освітлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2.007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

4. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

5. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення: 16.06.2023).

6. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99**.** – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

7. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

8. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення: 16.06.2023).

9. Методичні рекомендації до виконання розділу «Заходи з охорони праці та техніки безпеки» у магістерській дисертації / Л.Д. Третякова; М-во освіти і науки України, Національний технічний університет україни «Київський політехнічний інститут» – Київ, КПІ, 2014. – 26 с. – Режим доступу до ресурсу: http://surl.li/dhulo (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– Досліджена система платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

– На основі отриманих результатів досліджень створена програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 4217 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,24 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іщенко Д.А. Дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

3. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

4. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

5. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

6. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

7. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

8. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

9. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

10. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

11. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

12. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

13. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

14. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

15. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

16. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

17. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

18. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

19. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

20. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

21. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

22. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

23. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

25. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

26. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

27. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

28. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

31. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

32. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

33. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

34. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

35. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

36. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

37. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

38. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

39. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

40. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

41. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

43. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

44. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

45. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

46. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

47. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

48. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

49. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. – Випуск 2 (46) – Х.: ХУПС – 2016. – С. 146-149.

50. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Builder C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 95 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Кислун О.А.

Дослідження та програмна реалізація системи платформи управління мережними пристроями інтелектуального будинку за технологією CWMP

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2023 року

Основна програма

Файл Project_ihomeCWMP.cpp основної програми

```
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("main.cpp", Form_main);
USEFORM("about.cpp", Form_about);
USEFORM("light.cpp", Form_light);
USEFORM("water.cpp", Form_water);
USEFORM("gas.cpp", Form_gas);
USEFORM("security.cpp", Form_security);
USEFORM("climate.cpp", Form_climate);
USEFORM("phone.cpp", Form_phone);
USEFORM("tv.cpp", Form_tv);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm_main), &Form_main);
                Application->CreateForm(__classid(TForm_about), &Form_about);
                Application->CreateForm(__classid(TForm_light), &Form_light);
                Application->CreateForm(__classid(TForm_water), &Form_water);
                Application->CreateForm(__classid(TForm_gas), &Form_gas);
                Application->CreateForm(__classid(TForm_security),
&Form_security);
                Application->CreateForm(__classid(TForm_climate),
&Form_climate);
                Application->CreateForm(__classid(TForm_phone), &Form_phone);
                Application->CreateForm(__classid(TForm_tv), &Form_tv);

                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Файл main.cpp основної програми

```
//---------------------------------------------------------------------
//підключення бібліотек
#include <vcl.h>
#pragma hdrstop

//підключення модулів програми
#include "main.h"
#include "light.h"
#include "water.h"
#include "gas.h"
#include "security.h"
#include "climate.h"
#include "phone.h"
#include "tv.h"
#include "about.h"

//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_main *Form_main;
//---------------------------------------------------------------------
__fastcall TForm_main::TForm_main(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------

//відкриття вікна "Управляння освітленням"
void __fastcall TForm_main::Button1Click(TObject *Sender)
{
Form_light->Show();
}
//---------------------------------------------------------------------


//відкриття вікна "Про програму..."
void __fastcall TForm_main::Button8Click(TObject *Sender)
{
Form_about->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Система водопостачання"
void __fastcall TForm_main::Button4Click(TObject *Sender)
{
Form_water->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Система клімат-контролю"
void __fastcall TForm_main::Button2Click(TObject *Sender)
{
Form_climate->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Система газопостачання"
void __fastcall TForm_main::Button5Click(TObject *Sender)
{
Form_gas->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Система безпеки"
```

```
void __fastcall TForm_main::Button3Click(TObject *Sender)
{
Form_security->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Домашній кінотеатр"
void __fastcall TForm_main::Button7Click(TObject *Sender)
{
Form_tv->Show();
}
//---------------------------------------------------------------------

//відкриття вікна "Телефонній зв'язок"
void __fastcall TForm_main::Button6Click(TObject *Sender)
{
Form_phone->Show();
}
//---------------------------------------------------------------------
```

Файл main.h – бібліотека для файлу main.cpp

```
//---------------------------------------------------------------------
#ifndef mainH
#define mainH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <Buttons.hpp>
//---------------------------------------------------------------------
class TForm_main : public TForm
{
__published:      // IDE-managed Components
        TImage *Image1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TButton *Button6;
        TButton *Button7;
        TImage *Image2;
        TImage *Image3;
        TImage *Image4;
        TImage *Image5;
        TImage *Image6;
        TImage *Image7;
        TImage *Image8;
        TButton *Button8;
        TImage *Image9;
        TLabel *Label1;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
private:    // User declarations
public:         // User declarations
        __fastcall TForm_main(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm_main *Form_main;
//---------------------------------------------------------------------
#endif
```

Файл light.cpp – керування освітленням

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "light.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_light *Form_light;
//---------------------------------------------------------------------------
__fastcall TForm_light::TForm_light(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

//ввімкнення ліхтаря біля входу в будинок з вказаною яскравістю
void __fastcall TForm_light::torch_onClick(TObject *Sender)
{
torch->Caption="Ввімкнено";
Image_torch_on->Visible=true;
Image_torch_off->Visible=false;
TrackBar_torch->Enabled=true;

ClientSocket->Send_CWMP_Command(0,1,18,1);  //0-код будинку; 1-код пристрою, що
керує ліхтарем; 18-код команди "on"; 1-кількість повторних надсилань команди
ClientSocket->SendLeviton_CWMP_(0,1,TrackBar_torch->Position); //встановлення
яскравості, вказаної користувачем за допомогою TrackBar-у


}
//---------------------------------------------------------------------------
//вимкнення ліхтаря біля входу в будинок
void __fastcall TForm_light::torch_offClick(TObject *Sender)
{
torch->Caption="Вимкнено";
Image_torch_off->Visible=true;
Image_torch_on->Visible=false;
TrackBar_torch->Enabled=false;
ClientSocket->Send_CWMP_Command(0,1,19,1);  //0-код будинку; 1-код пристрою, що
керує ліхтарем; 19-код команди "off"; 1-кількість повторних надсилань команди

}
//---------------------------------------------------------------------------
//ввімкнення світла в коридорі з вказаною яскравістю
void __fastcall TForm_light::corridor_onClick(TObject *Sender)
{
corridor->Caption="Ввімкнено";
Image_corridor_on->Visible=true;
Image_corridor_off->Visible=false;
TrackBar_corridor->Enabled=true;

ClientSocket->Send_CWMP_Command(0,2,18,1);
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_corridor->Position);

}
//---------------------------------------------------------------------------

//ввімкнення світла у вітальні з вказаною яскравістю
void __fastcall TForm_light::drawing_room_onClick(TObject *Sender)
{
drawing_room->Caption="Ввімкнено";
Image_drawing_room_on->Visible=true;
```

```
Image_drawing_room_off->Visible=false;
TrackBar_drawing_room->Enabled=true;

ClientSocket->Send_CWMP_Command(0,3,18,1);
ClientSocket->SendLeviton_CWMP_(0,3,TrackBar_drawing_room->Position);
}
//---------------------------------------------------------------------------

//ввімкнення світла на кухні з вказаною яскравістю
void __fastcall TForm_light::kitchen_onClick(TObject *Sender)
{
kitchen->Caption="Ввімкнено";
Image_kitchen_on->Visible=true;
Image_kitchen_off->Visible=false;
TrackBar_kitchen->Enabled=true;

ClientSocket->Send_CWMP_Command(0,4,18,1);
ClientSocket->SendLeviton_CWMP_(0,4,TrackBar_kitchen->Position);
}
//---------------------------------------------------------------------------

//ввімкнення світла в кабінеті з вказаною яскравістю
void __fastcall TForm_light::cabinet_onClick(TObject *Sender)
{
cabinet->Caption="Ввімкнено";
Image_cabinet_on->Visible=true;
Image_cabinet_off->Visible=false;
TrackBar_cabinet->Enabled=true;

ClientSocket->Send_CWMP_Command(0,5,18,1);
ClientSocket->SendLeviton_CWMP_(0,5,TrackBar_cabinet->Position);
}
//---------------------------------------------------------------------------

//ввімкнення світла у спальні з вказаною яскравістю
void __fastcall TForm_light::bedroom_onClick(TObject *Sender)
{
bedroom->Caption="Ввімкнено";
Image_bedroom_on->Visible=true;
Image_bedroom_off->Visible=false;
TrackBar_bedroom->Enabled=true;

ClientSocket->Send_CWMP_Command(0,6,18,1);
ClientSocket->SendLeviton_CWMP_(0,6,TrackBar_bedroom->Position);
}
//---------------------------------------------------------------------------

//ввімкнення світла в дитячій кімнаті з вказаною яскравістю
void __fastcall TForm_light::baby_room_onClick(TObject *Sender)
{
baby_room->Caption="Ввімкнено";
Image_baby_room_on->Visible=true;
Image_baby_room_off->Visible=false;
TrackBar_baby_room->Enabled=true;

ClientSocket->Send_CWMP_Command(0,7,18,1);
ClientSocket->SendLeviton_CWMP_(0,7,TrackBar_baby_room->Position);
}
//---------------------------------------------------------------------------

//ввімкнення світла у ванній кімнаті з вказаною яскравістю
void __fastcall TForm_light::bathroom_onClick(TObject *Sender)
{
bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
TrackBar_bathroom->Enabled=true;

ClientSocket->Send_CWMP_Command(0,8,18,1);
```

```
ClientSocket->SendLeviton_CWMP_(0,8,TrackBar_bathroom->Position);
}


//---------------------------------------------------------------------------

//вимкнення світла в коридорі
void __fastcall TForm_light::corridor_offClick(TObject *Sender)
{
corridor->Caption="Вимкнено";
Image_corridor_off->Visible=true;
Image_corridor_on->Visible=false;
TrackBar_corridor->Enabled=false;
ClientSocket->Send_CWMP_Command(0,2,19,1);
}
//---------------------------------------------------------------------------

//вимкнення світла у вітальні
void __fastcall TForm_light::drawing_room_offClick(TObject *Sender)
{
drawing_room->Caption="Вимкнено";
Image_drawing_room_off->Visible=true;
Image_drawing_room_on->Visible=false;
TrackBar_drawing_room->Enabled=false;
ClientSocket->Send_CWMP_Command(0,3,19,1);
}
//---------------------------------------------------------------------------

//вимкнення світла на кухні
void __fastcall TForm_light::kitchen_offClick(TObject *Sender)
{
kitchen->Caption="Вимкнено";
Image_kitchen_off->Visible=true;
Image_kitchen_on->Visible=false;
TrackBar_kitchen->Enabled=false;
ClientSocket->Send_CWMP_Command(0,4,19,1);
}
//---------------------------------------------------------------------------

//вимкнення світла в кабінеті
void __fastcall TForm_light::cabinet_offClick(TObject *Sender)
{
cabinet->Caption="Вимкнено";
Image_cabinet_off->Visible=true;
Image_cabinet_on->Visible=false;
TrackBar_cabinet->Enabled=false;
ClientSocket->Send_CWMP_Command(0,5,19,1);
}
//---------------------------------------------------------------------------

//вимкнення світла у спальні
void __fastcall TForm_light::bedroom_offClick(TObject *Sender)
{
bedroom->Caption="Вимкнено";
Image_bedroom_off->Visible=true;
Image_bedroom_on->Visible=false;
TrackBar_bedroom->Enabled=false;
ClientSocket->Send_CWMP_Command(0,6,19,1);
}
//---------------------------------------------------------------------------

//вимкнення світла у дитячій кімнаті
void __fastcall TForm_light::baby_room_offClick(TObject *Sender)
{
baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
TrackBar_baby_room->Enabled=false;
ClientSocket->Send_CWMP_Command(0,7,19,1);
```

```
}
//---------------------------------------------------------------------------

//вимкнення світла у ванній кімнаті
void __fastcall TForm_light::bathroom_offClick(TObject *Sender)
{
bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
TrackBar_bathroom->Enabled=false;
ClientSocket->Send_CWMP_Command(0,8,19,1);
}
//---------------------------------------------------------------------------

//ввімкнення світла скрізь
void __fastcall TForm_light::Button18Click(TObject *Sender)
{

torch->Caption="Ввімкнено";
Image_torch_on->Visible=true;
Image_torch_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,1,18,1);
ClientSocket->SendLeviton_CWMP_(0,1,TrackBar_torch->Position);

corridor->Caption="Ввімкнено";
Image_corridor_on->Visible=true;
Image_corridor_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,2,18,1);
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_corridor->Position);

drawing_room->Caption="Ввімкнено";
Image_drawing_room_on->Visible=true;
Image_drawing_room_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,3,18,1);
ClientSocket->SendLeviton_CWMP_(0,3,TrackBar_drawing_room->Position);

kitchen->Caption="Ввімкнено";
Image_kitchen_on->Visible=true;
Image_kitchen_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,4,18,1);
ClientSocket->SendLeviton_CWMP_(0,4,TrackBar_kitchen->Position);

cabinet->Caption="Ввімкнено";
Image_cabinet_on->Visible=true;
Image_cabinet_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,5,18,1);
ClientSocket->SendLeviton_CWMP_(0,5,TrackBar_cabinet->Position);

bedroom->Caption="Ввімкнено";
Image_bedroom_on->Visible=true;
Image_bedroom_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,6,18,1);
ClientSocket->SendLeviton_CWMP_(0,6,TrackBar_bedroom->Position);

baby_room->Caption="Ввімкнено";
Image_baby_room_on->Visible=true;
Image_baby_room_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,7,18,1);
ClientSocket->SendLeviton_CWMP_(0,7,TrackBar_baby_room->Position);


bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
ClientSocket->Send_CWMP_Command(0,8,18,1);
ClientSocket->SendLeviton_CWMP_(0,8,TrackBar_bathroom->Position);


TrackBar_torch->Enabled=true;
```

```
TrackBar_bedroom->Enabled=true;
TrackBar_corridor->Enabled=true;
TrackBar_drawing_room->Enabled=true;
TrackBar_kitchen->Enabled=true;
TrackBar_cabinet->Enabled=true;
TrackBar_baby_room->Enabled=true;
TrackBar_bathroom->Enabled=true;
}
//---------------------------------------------------------------------------

//вимкнення світла скрізь
void __fastcall TForm_light::Button17Click(TObject *Sender)
{
torch->Caption="Вимкнено";
Image_torch_off->Visible=true;
Image_torch_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,1,19,1);

corridor->Caption="Вимкнено";
Image_corridor_off->Visible=true;
Image_corridor_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,2,19,1);

drawing_room->Caption="Вимкнено";
Image_drawing_room_off->Visible=true;
Image_drawing_room_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,3,19,1);

kitchen->Caption="Вимкнено";
Image_kitchen_off->Visible=true;
Image_kitchen_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,4,19,1);

cabinet->Caption="Вимкнено";
Image_cabinet_off->Visible=true;
Image_cabinet_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,5,19,1);

bedroom->Caption="Вимкнено";
Image_bedroom_off->Visible=true;
Image_bedroom_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,6,19,1);

baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,7,19,1);


bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
ClientSocket->Send_CWMP_Command(0,8,19,1);

TrackBar_torch->Enabled=false;
TrackBar_bedroom->Enabled=false;
TrackBar_corridor->Enabled=false;
TrackBar_drawing_room->Enabled=false;
TrackBar_kitchen->Enabled=false;
TrackBar_cabinet->Enabled=false;
TrackBar_baby_room->Enabled=false;
TrackBar_bathroom->Enabled=false;
}
//---------------------------------------------------------------------------

//імітація присутності господарів
//ввімкнення та вимкнення світла випадковим чином
void __fastcall TForm_light::Button1Click(TObject *Sender)
{
```

```
if(Button1->Caption=="Імітація присутності")
{
Timer1->Enabled=true;      //затуск таймеру, що запрограмований на імітацію
присутності
Button1->Caption=="Вимкнути імітацію"
}
else
{
Timer1->Enabled=false;     //зупинтка таймеру
Button1->Caption=="Імітація присутності"
}

}
//---------------------------------------------------------------------------

//таймер, запрограмований на імітацію присутності
void __fastcall TForm_light::Timer1Timer(TObject *Sender)
{
int x, y;
randomize();
x=random (6)+2;  //генерація випадкового номера лампи в діапазоні 2-8
ClientSocket->Send_CWMP_Command(0,x,18,1);
y=random (6)+2;
ClientSocket->Send_CWMP_Command(0,y,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_light::brightnessTimer(TObject *Sender)
{
//зміна яскравості
}
//---------------------------------------------------------------------------

//зміна яскравості ліхтаря
void __fastcall TForm_light::TrackBar_torchChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,1,TrackBar_torch->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення коридору
void __fastcall TForm_light::TrackBar_corridorChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_corridor->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення вітальні
void __fastcall TForm_light::TrackBar_drawing_roomChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_drawing_room->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення кухні
void __fastcall TForm_light::TrackBar_kitchenChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_kitchen->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення кабінету
void __fastcall TForm_light::TrackBar_cabinetChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_cabinet->Position);
}
//---------------------------------------------------------------------------
```

```
//зміна яскравості освітлення спальні
void __fastcall TForm_light::TrackBar_bedroomChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_bedroom->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення дитячої
void __fastcall TForm_light::TrackBar_baby_roomChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_baby_room->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення ванної
void __fastcall TForm_light::TrackBar_bathroomChange(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,2,TrackBar_bathroom->Position);
}
//---------------------------------------------------------------------------

//визначення та виведення на екран поточного стану освітлення
void __fastcall TForm_light::StatusTimer(TObject *Sender)
{
ShortString st;

ClientSocket->Send_CWMP_StatusQuery();

st=_CWMP_State->GetStateOnOff(0, 1);
if(st="On") {
torch->Caption="Ввімкнено";
Image_torch_on->Visible=true;
Image_torch_off->Visible=false;
TrackBar_torch->Enabled=true;
}
else {
torch->Caption="Вимкнено";
Image_torch_off->Visible=true;
Image_torch_on->Visible=false;
TrackBar_torch->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 2);
if(st="On") {
corridor->Caption="Ввімкнено";
Image_corridor_on->Visible=true;
Image_corridor_off->Visible=false;
TrackBar_corridor->Enabled=true;
}
else {
corridor->Caption="Вимкнено";
Image_corridor_off->Visible=true;
Image_corridor_on->Visible=false;
TrackBar_corridor->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 3);
if(st="On") {
drawing_room->Caption="Ввімкнено";
Image_drawing_room_on->Visible=true;
Image_drawing_room_off->Visible=false;
TrackBar_drawing_room->Enabled=true;
}
else {
drawing_room->Caption="Вимкнено";
Image_drawing_room_off->Visible=true;
Image_drawing_room_on->Visible=false;
TrackBar_drawing_room->Enabled=false;
}
```

```
st=_CWMP_State->GetStateOnOff(0, 4);
if(st="On") {
kitchen->Caption="Ввімкнено";
Image_kitchen_on->Visible=true;
Image_kitchen_off->Visible=false;
TrackBar_kitchen->Enabled=true;
}
else {
kitchen->Caption="Вимкнено";
Image_kitchen_off->Visible=true;
Image_kitchen_on->Visible=false;
TrackBar_kitchen->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 5);
if(st="On") {
cabinet->Caption="Ввімкнено";
Image_cabinet_on->Visible=true;
Image_cabinet_off->Visible=false;
TrackBar_cabinet->Enabled=true;
}
else {
cabinet->Caption="Вимкнено";
Image_cabinet_off->Visible=true;
Image_cabinet_on->Visible=false;
TrackBar_cabinet->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 6);
if(st="On") {
bedroom->Caption="Ввімкнено";
Image_bedroom_on->Visible=true;
Image_bedroom_off->Visible=false;
TrackBar_bedroom->Enabled=true;
}
else {
bedroom->Caption="Вимкнено";
Image_bedroom_off->Visible=true;
Image_bedroom_on->Visible=false;
TrackBar_bedroom->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 7);
if(st="On") {
baby_room->Caption="Ввімкнено";
Image_baby_room_on->Visible=true;
Image_baby_room_off->Visible=false;
TrackBar_baby_room->Enabled=true;
}
else {
baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
TrackBar_baby_room->Enabled=false;
}

st=_CWMP_State->GetStateOnOff(0, 8);
if(st="On") {
bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
TrackBar_bathroom->Enabled=true;
}
else {
bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
TrackBar_bathroom->Enabled=false;
```

```
}
}
```

Файл light.h – бібліотека для файлу light.cpp

```
#ifndef lightH
#define lightH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm_light : public TForm
{
__published:      // IDE-managed Components
        TTrackBar *TrackBar_corridor;
        TLabel *Label11;
        TImage *Image_torch_on;
        TImage *Image_torch_off;
        TTrackBar *TrackBar_torch;
        TLabel *Label8;
        TImage *Image_corridor_on;
        TImage *Image_corridor_off;
        TTrackBar *TrackBar_drawing_room;
        TLabel *Label9;
        TImage *Image_drawing_room_on;
        TImage *Image_drawing_room_off;
        TTrackBar *TrackBar_cabinet;
        TLabel *Label12;
        TImage *Image_cabinet_on;
        TImage *Image_cabinet_off;
        TTrackBar *TrackBar_bedroom;
        TLabel *Label13;
        TImage *Image_bedroom_on;
        TImage *Image_bedroom_off;
        TTrackBar *TrackBar_baby_room;
        TLabel *Label14;
        TImage *Image_baby_room_on;
        TImage *Image_baby_room_off;
        TTrackBar *TrackBar_kitchen;
        TLabel *Label15;
        TImage *Image_kitchen_on;
        TImage *Image_kitchen_off;
        TTrackBar *TrackBar_bathroom;
        TLabel *Label16;
        TImage *Image_bathroom_on;
        TImage *Image_bathroom_off;
        TButton *Button17;
        TButton *Button18;
        TLabel *Label17;
        TLabel *torch;
        TLabel *Label19;
        TLabel *corridor;
        TLabel *Label21;
        TLabel *drawing_room;
        TLabel *Label23;
        TLabel *kitchen;
        TLabel *Label25;
        TLabel *cabinet;
        TLabel *Label27;
        TLabel *bedroom;
        TLabel *Label29;
        TLabel *baby_room;
        TLabel *Label31;
        TLabel *bathroom;
        TBevel *Bevel1;
```

```
        TLabel *Label10;
        TBevel *Bevel2;
        TLabel *Label1;
        TBevel *Bevel3;
        TBevel *Bevel4;
        TBevel *Bevel5;
        TBevel *Bevel6;
        TBevel *Bevel7;
        TBevel *Bevel8;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label7;
        TLabel *Label2;
        TLabel *Label6;
        TLabel *Label5;
        TButton *torch_on;
        TButton *torch_off;
        TButton *corridor_on;
        TButton *corridor_off;
        TButton *drawing_room_on;
        TButton *drawing_room_off;
        TButton *kitchen_on;
        TButton *kitchen_off;
        TButton *cabinet_on;
        TButton *cabinet_off;
        TButton *bedroom_on;
        TButton *bedroom_off;
        TButton *baby_room_on;
        TButton *baby_room_off;
        TButton *bathroom_on;
        TButton *bathroom_off;
        TButton *Button1;
        TTimer *Timer1;
        TTimer *Status;
        void __fastcall torch_onClick(TObject *Sender);
        void __fastcall torch_offClick(TObject *Sender);
        void __fastcall corridor_onClick(TObject *Sender);
        void __fastcall drawing_room_onClick(TObject *Sender);
        void __fastcall kitchen_onClick(TObject *Sender);
        void __fastcall cabinet_onClick(TObject *Sender);
        void __fastcall bedroom_onClick(TObject *Sender);
        void __fastcall baby_room_onClick(TObject *Sender);
        void __fastcall bathroom_onClick(TObject *Sender);
        void __fastcall corridor_offClick(TObject *Sender);
        void __fastcall drawing_room_offClick(TObject *Sender);
        void __fastcall kitchen_offClick(TObject *Sender);
        void __fastcall cabinet_offClick(TObject *Sender);
        void __fastcall bedroom_offClick(TObject *Sender);
        void __fastcall baby_room_offClick(TObject *Sender);
        void __fastcall bathroom_offClick(TObject *Sender);
        void __fastcall Button18Click(TObject *Sender);
        void __fastcall Button17Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall brightnessTimer(TObject *Sender);
        void __fastcall TrackBar_torchChange(TObject *Sender);
        void __fastcall TrackBar_corridorChange(TObject *Sender);
        void __fastcall TrackBar_drawing_roomChange(TObject *Sender);
        void __fastcall TrackBar_kitchenChange(TObject *Sender);
        void __fastcall TrackBar_cabinetChange(TObject *Sender);
        void __fastcall TrackBar_bedroomChange(TObject *Sender);
        void __fastcall TrackBar_baby_roomChange(TObject *Sender);
        void __fastcall TrackBar_bathroomChange(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_light(TComponent* Owner);
};
//---------------------------------------------------------------------------
```

```
extern PACKAGE TForm_light *Form_light;
//---------------------------------------------------------------------------
#endif
```

Файл climate.cpp – керування кліматом

```
#include <vcl.h>
#pragma hdrstop

#include "climate.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_climate *Form_climate;
//---------------------------------------------------------------------------
__fastcall TForm_climate::TForm_climate(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_climate::Button3Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,21,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button4Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,21,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button1Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,22,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button2Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,22,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button5Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,23,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button6Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,23,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button7Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,24,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button8Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,24,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button9Click(TObject *Sender)
{
```

```
ClientSocket->Send_CWMP_Command(0,26,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button10Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,26,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button11Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,27,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button12Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,27,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button13Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,28,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button14Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,28,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button15Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,29,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button16Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,29,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar1Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,21,TrackBar1->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar2Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,22,TrackBar2->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar3Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,23,TrackBar3->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar4Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,24,TrackBar4->Position);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm_climate::TrackBar5Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,24,TrackBar5->Position);
}
//---------------------------------------------------------------------------



//визначення та виведення на екран поточного стану клімат-контролю

void __fastcall TForm_climate::StatusTimer(TObject *Sender)
{
ShortString st;
int n;

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 21);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 22);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 23);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 24);
if(st="On") Label_4->Caption="Ввімкнено"; else Label_4->Caption="Вимкнено";

n=GetVariable(25);
Label_t->Caption=n;
n=GetVariable(34);
Label_v->Caption=n;

n=GetVariable(22);
Label_tp1->Caption=n;
n=GetVariable(23);
Label_tp2->Caption=n;

st=_CWMP_State->GetStateOnOff(0, 26);
if(st="On") Label_5->Caption="Відкрито"; else Label_5->Caption="Закрито";
st=_CWMP_State->GetStateOnOff(0, 27);
if(st="On") Label_6->Caption="Відкрито"; else Label_6->Caption="Закрито";
st=_CWMP_State->GetStateOnOff(0, 28);
if(st="On") Label_7->Caption="Відкрито"; else Label_7->Caption="Закрито";
st=_CWMP_State->GetStateOnOff(0, 29);
if(st="On") Label_8->Caption="Відкрито"; else Label_8->Caption="Закрито";
}
```

Файл climate.h – бібліотека для файлу climate.cpp

```
#ifndef climateH
#define climateH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------
class TForm_climate : public TForm
{
__published:      // IDE-managed Components
        TGroupBox *GroupBox2;
        TLabel *Label3;
        TLabel *Label_1;
        TButton *Button3;
        TButton *Button4;
        TTrackBar *TrackBar1;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label7;
        TGroupBox *GroupBox1;
        TLabel *Label1;
        TLabel *Label_2;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *Label16;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label27;
        TLabel *Label_tp1;
        TLabel *Label29;
        TLabel *Label30;
        TBevel *Bevel2;
        TButton *Button1;
        TButton *Button2;
        TTrackBar *TrackBar2;
        TGroupBox *GroupBox3;
        TLabel *Label31;
        TLabel *Label_3;
        TLabel *Label33;
        TLabel *Label34;
        TLabel *Label35;
        TLabel *Label36;
        TLabel *Label37;
        TLabel *Label38;
        TLabel *Label39;
        TLabel *Label40;
        TLabel *Label41;
        TLabel *Label42;
        TLabel *Label_tp2;
        TLabel *Label44;
        TLabel *Label45;
```

```
TBevel *Bevel3;
TButton *Button5;
TButton *Button6;
TTrackBar *TrackBar3;
TGroupBox *GroupBox4;
TLabel *Label46;
TLabel *Label_4;
TLabel *Label48;
TLabel *Label49;
TButton *Button7;
TButton *Button8;
TGroupBox *GroupBox5;
TLabel *Label57;
TLabel *Label_5;
TButton *Button9;
TButton *Button10;
TGroupBox *GroupBox6;
TLabel *Label59;
TLabel *Label_6;
TButton *Button11;
TButton *Button12;
TGroupBox *GroupBox7;
TLabel *Label61;
TLabel *Label_7;
TButton *Button13;
TButton *Button14;
TGroupBox *GroupBox8;
TLabel *Label63;
TLabel *Label_8;
TButton *Button15;
TButton *Button16;
TGroupBox *GroupBox9;
TLabel *Label67;
TLabel *Label68;
TLabel *Label_t;
TLabel *Label70;
TLabel *Label71;
TLabel *Label_v;
TImage *Image8;
TImage *Image7;
TTrackBar *TrackBar4;
TLabel *Label66;
TLabel *Label73;
TLabel *Label74;
TLabel *Label75;
TLabel *Label76;
TLabel *Label77;
TLabel *Label78;
TLabel *Label79;
TLabel *Label65;
TLabel *Label23;
TTrackBar *TrackBar5;
TLabel *Label24;
TLabel *Label25;
TLabel *Label26;
TLabel *Label50;
TLabel *Label51;
TLabel *Label52;
TLabel *Label53;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label80;
TLabel *Label81;
TButton *Button17;
TLabel *Label82;
TLabel *Label83;
TEdit *Edit1;
TLabel *Label84;
```

```
        TEdit *Edit2;
        TLabel *Label85;
        TTimer *Status;
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button9Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
        void __fastcall Button11Click(TObject *Sender);
        void __fastcall Button12Click(TObject *Sender);
        void __fastcall Button13Click(TObject *Sender);
        void __fastcall Button14Click(TObject *Sender);
        void __fastcall Button15Click(TObject *Sender);
        void __fastcall Button16Click(TObject *Sender);
        void __fastcall TrackBar1Change(TObject *Sender);
        void __fastcall TrackBar2Change(TObject *Sender);
        void __fastcall TrackBar3Change(TObject *Sender);
        void __fastcall TrackBar4Change(TObject *Sender);
        void __fastcall TrackBar5Change(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_climate(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm_climate *Form_climate;
//---------------------------------------------------------------------
#endif
```

Файл water.cpp – керування системою водопостачання

```
#include <vcl.h>
#pragma hdrstop

#include "water.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_water *Form_water;
//---------------------------------------------------------------------------
__fastcall TForm_water::TForm_water(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
//ввімкнення системи контролю протікання води у ванній
void __fastcall TForm_water::Button2Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,9,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану холодної води у ванні
void __fastcall TForm_water::Button6Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,10,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану гарячої води у ванні
void __fastcall TForm_water::Button8Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,11,18,1);
}
//---------------------------------------------------------------------------
//ввімкнення системи контролю протікання води на кухні
void __fastcall TForm_water::Button4Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,12,18,1);
}
//---------------------------------------------------------------------------
//відкриття клапану холодної води на кухні
void __fastcall TForm_water::Button10Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,13,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану гарячої води на кухні
void __fastcall TForm_water::Button12Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,14,18,1);
}
//---------------------------------------------------------------------------

//вимкнення системи контролю протікання води у ванній
void __fastcall TForm_water::Button1Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,9,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану холодної води у ванні
void __fastcall TForm_water::Button5Click(TObject *Sender)
{
```

```
ClientSocket->Send_CWMP_Command(0,10,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану гарячої води у ванні
void __fastcall TForm_water::Button7Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,11,19,1);
}
//---------------------------------------------------------------------------

//вимкнення системи контролю протікання води на кухні
void __fastcall TForm_water::Button3Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,12,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану холодної води на кухні
void __fastcall TForm_water::Button9Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,13,19,1);
}
//---------------------------------------------------------------------------

//закриття клапану гарячої води на кухні
void __fastcall TForm_water::Button11Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,14,19,1);
}
//визначення та виведення на екран поточного стану системи водопостачання

void __fastcall TForm_water::StatusTimer(TObject *Sender)
{
ShortString st;
int n;

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 9);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 12);
if(st="On") Label_4->Caption="Ввімкнено"; else Label_4->Caption="Вимкнено";

n=GetVariable(9);
if(n="0") Label_11->Caption="В нормі"; else Label_11->Caption="Протікання води";
n=GetVariable(12);
if(n="0") Label_44->Caption="В нормі"; else Label_44->Caption="Протікання води";

st=_CWMP_State->GetStateOnOff(0, 10);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 11);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";

st=_CWMP_State->GetStateOnOff(0, 13);
if(st="On") Label_5->Caption="Ввімкнено"; else Label_5->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 14);
if(st="On") Label_6->Caption="Ввімкнено"; else Label_6->Caption="Вимкнено";
}
```

Файл water.h – бібліотека для файлу water.cpp

```
#ifndef waterH
#define waterH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm_water : public TForm
{
__published:      // IDE-managed Components
        TGroupBox *GroupBox1;
        TButton *Button1;
        TButton *Button2;
        TLabel *Label1;
        TLabel *Label_1;
        TGroupBox *GroupBox2;
        TLabel *Label3;
        TLabel *Label_4;
        TButton *Button3;
        TButton *Button4;
        TGroupBox *GroupBox3;
        TLabel *Label5;
        TLabel *Label_2;
        TButton *Button5;
        TButton *Button6;
        TGroupBox *GroupBox4;
        TLabel *Label7;
        TLabel *Label_3;
        TButton *Button7;
        TButton *Button8;
        TGroupBox *GroupBox5;
        TLabel *Label9;
        TLabel *Label_5;
        TButton *Button9;
        TButton *Button10;
        TGroupBox *GroupBox6;
        TLabel *Label11;
        TLabel *Label_6;
        TButton *Button11;
        TButton *Button12;
        TLabel *Label13;
        TLabel *Label_11;
        TLabel *Label15;
        TLabel *Label_44;
        TTimer *Status;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
        void __fastcall Button12Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button9Click(TObject *Sender);
        void __fastcall Button11Click(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_water(TComponent* Owner);
};
```

```
//---------------------------------------------------------------------------
extern PACKAGE TForm_water *Form_water;
//---------------------------------------------------------------------------
#endif
```

Файл gas.cpp – керування системою газопостачання

```
#include <vcl.h>
#pragma hdrstop

#include "gas.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_gas *Form_gas;
//---------------------------------------------------------------------------
__fastcall TForm_gas::TForm_gas(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_gas::Button2Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,15,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button1Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,15,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button4Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,16,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button3Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,16,19,1);
}
//---------------------------------------------------------------------------
//визначення та виведення на екран поточного стану системи газопостачання
void __fastcall TForm_gas::StatusTimer(TObject *Sender)
{
ShortString st;
int n;

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 15);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 16);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";

n=GetVariable(15);
if(n="0") Label_11->Caption="В нормі"; else Label_11->Caption="Виявлено витік
газу";
n=GetVariable(16);
if(n="0") Label_22->Caption="В нормі"; else Label_22->Caption="Виявлено дим";

}
```

Файл gas.h – бібліотека для файлу gas.cpp

```
#ifndef gasH
#define gasH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm_gas : public TForm
{
__published:      // IDE-managed Components
        TGroupBox *GroupBox1;
        TLabel *Label1;
        TLabel *Label_1;
        TLabel *Label13;
        TLabel *Label_11;
        TButton *Button1;
        TButton *Button2;
        TGroupBox *GroupBox2;
        TLabel *Label3;
        TLabel *Label_2;
        TLabel *Label5;
        TLabel *Label_22;
        TButton *Button3;
        TButton *Button4;
        TTimer *Status;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_gas(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_gas *Form_gas;
//---------------------------------------------------------------------------
#endif
```

Файл tv.cpp - керування дломашнім кінотеатром

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "tv.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_tv *Form_tv;
//---------------------------------------------------------------------------
__fastcall TForm_tv::TForm_tv(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_tv::TrackBar1Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,32,TrackBar1->Position);
}
//---------------------------------------------------------------------------


void __fastcall TForm_tv::onClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,32,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_tv::offClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,32,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_tv::ComboBox1Change(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,33,23,ComboBox1->Text);
}
//---------------------------------------------------------------------------
//визначення та виведення стану телевізору

void __fastcall TForm_tv::StatusTimer(TObject *Sender)
{
ShortString st;
int n;

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 33);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";

n=GetVariable(33);
ComboBox1->Text=n;

}
//---------------------------------------------------------------------------
```

Файл tv.h – бібліотека для файлу tv.cpp

```
#ifndef tvH
#define tvH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm_tv : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label11;
        TLabel *Label_1;
        TLabel *torch;
        TTrackBar *TrackBar1;
        TButton *on;
        TButton *off;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label4;
        TComboBox *ComboBox1;
        TImage *Image4;
        TBevel *Bevel1;
        TLabel *Label3;
        TTimer *Status;
        void __fastcall TrackBar1Change(TObject *Sender);
        void __fastcall onClick(TObject *Sender);
        void __fastcall offClick(TObject *Sender);
        void __fastcall ComboBox1Change(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_tv(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_tv *Form_tv;
//---------------------------------------------------------------------------
#endif
```

Файл phone.cpp – керування телефонним зв'язком

```
#include <vcl.h>
#pragma hdrstop

#include "phone.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_phone *Form_phone;
//---------------------------------------------------------------------------
__fastcall TForm_phone::TForm_phone(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::TrackBar1Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,30,TrackBar1->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::TrackBar2Change(TObject *Sender)
{
ClientSocket->SendLeviton_CWMP_(0,31,TrackBar2->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::t_onClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,30,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::t_offClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,30,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::a_onClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,31,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::a_offClick(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,31,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::Button3Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,30,23,Edit1->Text);
}
//---------------------------------------------------------------------------

//визначення та виведення стану телефонного зв'язку
void __fastcall TForm_phone::StatusTimer(TObject *Sender)
{
ShortString st;
int n;
```

```
ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 30);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 31);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";

}
//---------------------------------------------------------------------------
```

Файл phone.h – бібліотека для файлу phone.cpp

```
#ifndef phoneH
#define phoneH
//----------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//----------------------------------------------------------------------
class TForm_phone : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label11;
        TLabel *Label17;
        TLabel *Label_1;
        TTrackBar *TrackBar1;
        TButton *t_on;
        TButton *t_off;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label4;
        TEdit *Edit1;
        TButton *a_on;
        TButton *a_off;
        TLabel *Label6;
        TTrackBar *TrackBar2;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label_2;
        TButton *Button3;
        TImage *Image4;
        TImage *Image1;
        TBevel *Bevel1;
        TLabel *Label3;
        TBevel *Bevel2;
        TLabel *Label5;
        TTimer *Status;
        void __fastcall TrackBar1Change(TObject *Sender);
        void __fastcall TrackBar2Change(TObject *Sender);
        void __fastcall t_onClick(TObject *Sender);
        void __fastcall t_offClick(TObject *Sender);
        void __fastcall a_onClick(TObject *Sender);
        void __fastcall a_offClick(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_phone(TComponent* Owner);
};
extern PACKAGE TForm_phone *Form_phone;
#endif
```

Файл security.cpp – керування системою безпеки

```
#include <vcl.h>
#pragma hdrstop

#include "security.h"
#include "CPU_CWMP_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TForm_security *Form_security;
//---------------------------------------------------------------------------
__fastcall TForm_security::TForm_security(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_security::Button14Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,17,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button20Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,18,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button16Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,19,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button18Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,19,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button13Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,15,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button19Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,16,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button15Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,17,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button17Click(TObject *Sender)
{
ClientSocket->Send_CWMP_Command(0,18,19,1);
}
//---------------------------------------------------------------------------

//визначення та виведення стану системи безпеки
void __fastcall TForm_security::StatusTimer(TObject *Sender)
{
ShortString st;
int n;

ClientSocket->Send_CWMP_StatusQuery();
st=_CWMP_State->GetStateOnOff(0, 17);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 18);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";

st=_CWMP_State->GetStateOnOff(0, 19);
```

```
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_CWMP_State->GetStateOnOff(0, 20);
if(st="On") Label_4->Caption="Заблоковано"; else Label_4->Caption="Відкрито";


n=GetVariable(17);
if(n="0") Label_11->Caption="В нормі"; else Label_11->Caption="Спрацьовування";
n=GetVariable(18);
if(n="0") Label_22->Caption="В нормі"; else Label_22->Caption="Спрацьовування";
}
```

Файл security.h – бібліотека для файлу security.cpp

```
#ifndef securityH
#define securityH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Mask.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------
class TForm_security : public TForm
{
__published:      // IDE-managed Components
        TButton *Button12;
        TGroupBox *GroupBox1;
        TLabel *Label4;
        TLabel *Label_1;
        TLabel *Label13;
        TLabel *Label_11;
        TButton *Button13;
        TButton *Button14;
        TGroupBox *GroupBox2;
        TLabel *Label6;
        TLabel *Label_3;
        TButton *Button15;
        TButton *Button16;
        TGroupBox *GroupBox3;
        TLabel *Label8;
        TLabel *Label_4;
        TButton *Button17;
        TButton *Button18;
        TGroupBox *GroupBox4;
        TLabel *Label10;
        TLabel *Label_2;
        TLabel *Label12;
        TLabel *Label_22;
        TButton *Button19;
        TButton *Button20;
        TTimer *Status;
        void __fastcall Button14Click(TObject *Sender);
        void __fastcall Button20Click(TObject *Sender);
        void __fastcall Button16Click(TObject *Sender);
        void __fastcall Button18Click(TObject *Sender);
        void __fastcall Button13Click(TObject *Sender);
        void __fastcall Button19Click(TObject *Sender);
        void __fastcall Button15Click(TObject *Sender);
        void __fastcall Button17Click(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
private: // User declarations
public:     // User declarations
        __fastcall TForm_security(TComponent* Owner);
};
extern PACKAGE TForm_security *Form_security;
#endif
```

Файл CPU_CWMP_Socket.cpp – протокол CWMP

```
//------- -------------------

#include <basepch.h>

#pragma hdrstop

#include "CPU_CWMP_Socket.h"

#pragma package(smart_init)
//------- -------------------
//

static inline void ValidCtrCheck(TCPU_CWMP_Socket *)
{
    new TCPU_CWMP_Socket(NULL);
}
//#include "OcelotStateUnit.cpp"
//------- -------------------
__fastcall TCPU_CWMP_Socket::TCPU_CWMP_Socket(TComponent* Owner)
    : TClientSocket(Owner)
{
_CWMP_StateChangeNum=new T_CWMP_StateChangeNum;
OcelotStateChangeNum=new TOcelotStateChangeNum;
_CWMP_State=new T_CWMP_State;
OcelotState=new TOcelotState;

//Початкова ініціалізація змінних
_CWMP_StateChangeNum->Num=0;
OcelotStateChangeNum->Num=0;
FAuth=1;

// установлюємо номер порту
Port=63336;
SetVersion("");
OnConnect=MyOnConnect;
OnRead=MyOnRead;
OnDisconnect=MyOnDisconnect;
OnError=MyOnError;

}

__fastcall TCPU_CWMP_Socket::~TCPU_CWMP_Socket(void)
{
delete _CWMP_StateChangeNum;
delete OcelotStateChangeNum;
delete _CWMP_State;
delete OcelotState;

}
//------- -------------------
namespace CPU_CWMP_Socket
{
    void __fastcall PACKAGE Register()
    {
         TComponentClass classes[1] = {__classid(TCPU_CWMP_Socket)};
         RegisterComponents(" CPU-XA", classes, 0);
    }
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::MyOnConnect(System::TObject* Sender,
TCustomWinSocket* Socket)
{
char buf[16];
buf[0]='a';
memcpy(&buf[1],(Login).c_str(),5);
memcpy(&buf[6],(Password).c_str(),10);
SendBuf(buf,16);
```

```
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::MyOnDisconnect(System::TObject* Sender,
TCustomWinSocket* Socket)
{
SendBuf("d",1);
FAuth=2;
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,FAuth);
}

//------- -------------------
void __fastcall TCPU_CWMP_Socket::MyOnError(System::TObject* Sender,
TCustomWinSocket* Socket, TErrorEvent ErrorEvent, int &ErrorCode)
{
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,5);
if (OnMyError)
    OnMyError(Sender,Socket,ErrorEvent,ErrorCode);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::MyOnRead(System::TObject* Sender,
TCustomWinSocket* Socket)
{

long size=Socket->ReceiveLength();
char *buff=(char *)malloc(size);
char *local_pointer; // локальний покажчик на місце в буфері з якого починається
команда
long Offset=0; // зсув про буферу, що прийшов
long CommandStrSize=0; // розмір що прийшов команди

Socket->ReceiveBuf(buff,size);
local_pointer=&buff[Offset];

if (size==0) goto end_label; // якщо нічого не прийшло, то про всякий випадок
чистимо буфер

NewLoop:

// Перевіряємо на те, що це дійсно команда, а не обривок якого-небудь логу
if ((( size-Offset)>5)&&(memcmp(local_pointer,"CPUXA",5)==0))
    {
    Offset=Offset+5;
    local_pointer=&buff[Offset];
    }
else
    {
    goto end_label;
    }

if (( size-Offset)<2)
    goto end_label;
else
    {
    memcpy(&CommandStrSize,local_pointer,2); // Довідалися довжину текстової
команди
    Offset=Offset+2;
    local_pointer=&buff[Offset];
    }
if (( size-Offset)<CommandStrSize) goto end_label; // Якщо шматок до кінця
залишився менше, ніж довжина команди, виходимо з функції


switch ( local_pointer[0] )
{
case  'a':
            if (CommandStrSize==2)
                {
```

```
                FAuth=local_pointer[1];
                if (FOnChangeAuthStatus)
                    FOnChangeAuthStatus(this,FAuth);
                };
            break;

case  'c': break;

case  'd':
            if (CommandStrSize==2)
                {
                FAuth=2;
                if (FOnChangeAuthStatus) FOnChangeAuthStatus(this,FAuth);
                };
            break;

case  'p':
                if (FOnReadOtherData)
FOnReadOtherData(this,&local_pointer[0],CommandStrSize);

case  'q':
            if (CommandStrSize==1034+3)
                {
                memcpy(&OcelotStateChangeNum->Num,&local_pointer[1],2);
                OcelotState->LoadInfoBuff(&local_pointer[3]);
                if (FOnReadOcelotStatusData)
                    FOnReadOcelotStatusData(this,true);
                }
            if (CommandStrSize==3)
                if (FOnReadOcelotStatusData)
                    FOnReadOcelotStatusData(this,false);
            break;

case  'x':
            if (CommandStrSize==259)
                {
                memcpy(&_CWMP_StateChangeNum->Num,&local_pointer[1],2);
                _CWMP_State->LoadInfoBuff(&local_pointer[3]);
                if (FOnRead10StatusData)
                    FOnRead10StatusData(this,true);
                }
            if (CommandStrSize==3)
                if (FOnRead10StatusData)
                    FOnRead10StatusData(this,false);
            break;


default : if (FOnReadOtherData)
FOnReadOtherData(this,local_pointer,CommandStrSize);
}
if (( size-Offset-CommandStrSize)>0) // якщо шматок блоку, що залишився, більше
0 (може в купі лежить ще одна програма)
    {
    Offset=Offset+CommandStrSize; // указуємо зрушення
    local_pointer=&buff[Offset];
    goto NewLoop; // Пішли на нове коло
    }

end_label:
free(buff);
}
//------- ------------------
bool __fastcall TCPU_CWMP_Socket::SendBuf(char *buf,long size)
{
if (Active)
    {
    char *buff;
    long size_buff=size+7;
    buff=(char *)malloc(size_buff);
```

```
    memcpy(buff,"CPUXA",5);
    memcpy(&buff[5],&size,2);
    memcpy(&buff[7],buf,size);
    Socket->SendBuf(buff,size_buff);
    free(buff);
    return true;
    }
else
    return false;
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::SendOcelotStatusQuery(void)
{
char buf[3];
buf[0]='q';
memcpy(&buf[1],&OcelotStateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::Send10StatusQuery(void)
{
char buf[3];
buf[0]='x';
memcpy(&buf[1],&_CWMP_StateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
unsigned short __fastcall TCPU_CWMP_Socket::GetDataFromUnit(unsigned char
UnitNumber)
{
return OcelotState->GetDataFromUnit(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_CWMP_Socket::GetIO(unsigned char
UnitNumber,unsigned char Point)
{
return OcelotState->GetIO(UnitNumber,Point);
}
//------- -------------------
unsigned char __fastcall TCPU_CWMP_Socket::GetUnitVer(unsigned char UnitNumber)
{
return OcelotState->GetUnitVer(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_CWMP_Socket::GetUnitType(unsigned char UnitNumber)
{
return OcelotState->GetUnitType(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_CWMP_Socket::GetUnitsCount(void)
{
return OcelotState->GetUnitsCount();
}
//------- -------------------
TDateTime __fastcall TCPU_CWMP_Socket::GetCPUXADateTime()
{
return OcelotState->GetCPUXADateTime();
}
//------- -------------------
unsigned short __fastcall TCPU_CWMP_Socket::GetVariable(unsigned char VarNumber)
{
return OcelotState->GetVariable(VarNumber);
}
//------- -------------------
unsigned short __fastcall TCPU_CWMP_Socket::GetTimer(unsigned char TimerNumber)
{
return OcelotState->GetTimer(TimerNumber);
}
//------- -------------------
```

```
TDateTime TCPU_CWMP_Socket::GetTimerAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetTimer(VarNumber));
}
//------- -------------------
TDateTime TCPU_CWMP_Socket::GetVarAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetVariable(VarNumber));
}
//------- -------------------
TDateTime TCPU_CWMP_Socket::UShortToTime(unsigned short value)
{
  Word hour=div(value,100).quot;
  Word min=div(value,100).rem;
  try
  {
    return EncodeDateTime(1899, 12, 30, hour, min, 0, 0);
  }
  catch ( ... )
  { //12/30/1899 12:00 am
    return EncodeDateTime(1899, 12, 30, 12, 0, 0, 0);
  }
}
//------- -------------------
void TCPU_CWMP_Socket::SetTimeAsVar(unsigned char VarNumber, TDateTime time)
{
  SetVariable(VarNumber,TimeToUShort(time));
}
//------- -------------------
void TCPU_CWMP_Socket::SetTimeAsTimer(unsigned char VarNumber, TDateTime time)
{
  SetTimer(VarNumber,TimeToUShort(time));
}
//------- -------------------
unsigned short TCPU_CWMP_Socket::TimeToUShort(TDateTime time)
{
   Word Hour, Min, Sec, MSec;
   DecodeTime(time, Hour, Min, Sec, MSec);
   return Hour*100+Min;
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::SetIO(unsigned char UnitNumber,unsigned char
Point,unsigned char Stat)
{
char buf[5];
buf[0]='c';
buf[1]=0;
buf[2]=UnitNumber;
buf[3]=Point;
buf[4]=Stat;
SendBuf(buf,5);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket:: SetDateTime(unsigned char day,unsigned char
mon,unsigned char year,unsigned char hour,unsigned char min)
{
char buf[7];
buf[0]='c';
buf[1]=1;
buf[2]=day;
buf[3]=mon;
buf[4]=year;
buf[5]=hour;
buf[6]=min;
SendBuf(buf,7);

}
//------- -------------------
```

```
void __fastcall TCPU_CWMP_Socket::SetVariable(unsigned char VarNumber,unsigned
short Value)
{
char buf[5];
buf[0]='c';
buf[1]=2;
buf[2]=VarNumber;
memcpy(&buf[3],&Value,2);
SendBuf(buf,5);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::SetTimer(unsigned char TimerNumber,unsigned
short Value)
{
char buf[5];
buf[0]='c';
buf[1]=3;
buf[2]=TimerNumber;
memcpy(&buf[3],&Value,2);
SendBuf(buf,5);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::SendLeviton10(unsigned char house,unsigned
char key, unsigned char dim)
{
char buf[5];
buf[0]='c';
buf[1]=4;
buf[2]=house;
buf[3]=key;
buf[4]=dim;
SendBuf(buf,5);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::SendIr(unsigned short Value)
{
char buf[4];
buf[0]='c';
buf[1]=5;
memcpy(&buf[2],&Value,2);
SendBuf(buf,4);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::Send10Buff(unsigned char house,unsigned char
key, unsigned char repeat)
{
char buf[5];
buf[0]='c';
buf[1]=6;
buf[2]=house;
buf[3]=key;
buf[4]=repeat;
SendBuf(buf,5);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::Send10Command(unsigned char house,unsigned
char key, unsigned char CommandNum, unsigned char repeat)
{
char buf[6];
buf[0]='c';
buf[1]=7;
buf[2]=house;
buf[3]=key;
buf[4]=CommandNum;
buf[5]=repeat;
SendBuf(buf,6);
}
//------- -------------------
void __fastcall TCPU_CWMP_Socket::GetInternalProtocolVersion(void)
```

```
{
char buf[1];
buf[0]='p';
SendBuf(buf,1);
}
//------- -------------------

void __fastcall T_CWMP_StateChangeNum::Next(void)
{
if (Num==65534)
    Num=1;
else
    Num++;
};
//------- -------------------
__fastcall T_CWMP_StateChangeNum::T_CWMP_StateChangeNum(void)
{
Num=1;
};
//------- -------------------
AnsiString __fastcall T_CWMP_State::GetStateOnOff(unsigned char house, unsigned
char key)
{
if (map[house][key]==true)
return "On";
else
return " --";
}
//------- -------------------
bool __fastcall T_CWMP_State::GetOnOffStatus(unsigned char house, unsigned char
key)
{
if (map[house][key]==true)
return true;
else
return false;
}
//------- -------------------

bool __fastcall T_CWMP_State::LoadInfoBuff(char *buff)
{
memcpy(map,buff,256);
return true;
}
//------- -------------------

void _fastcall TCPU_CWMP_Socket::SetLogin(AnsiString str)
{
int len;
len=str.Length();
if (len<5)
    {
    for(int i=len;i<5;i++)
        str=str+"1";
    }
FLogin=str.SubString(1,5);
return;
}

void _fastcall TCPU_CWMP_Socket::SetPassword(AnsiString str)
{
int len;
len=str.Length();
if (len<10)
    for(int i=len;i<10;i++)
        str=str+"1";
FPassword=str.SubString(1,10);
return;
}
```

Файл CPU_CWMP_Socket.h - бібліотека для файлу CPU_CWMP_Socket.cpp

```
//------- ------------------

#ifndef CPU_CWMP_Socket
#define CPU_CWMP_Socket
//------- ------------------
#include <SysUtils.hpp>
#include <Classes.hpp>
#include <ScktComp.hpp>

#include "OcelotStateUnit.h"


//------- ------------------
const int MajorVersion = 1;
const int MinorVersion = 1;
// Опис типів викликуваних подій
    typedef void __fastcall (__closure *TOnReadOcelotStatusData)(System::TObject
*Sender, bool HaveNewData);
    typedef void __fastcall (__closure *TOnRead10StatusData)(System::TObject
*Sender, bool HaveNewData);
    typedef void __fastcall (__closure *TOnReadOtherData)(System::TObject
*Sender, char *Data, long size);
    typedef void __fastcall (__closure *TOnChangeAuthStatus)(System::TObject
*Sender, char FAuth);
    typedef void __fastcall (__closure *TOnMyError)(System::TObject* Sender,
TCustomWinSocket* Socket, TErrorEvent ErrorEvent, int &ErrorCode);

class T_CWMP_StateChangeNum {
public:           // User declarations
unsigned short Num;
void __fastcall Next(void);
__fastcall T_CWMP_StateChangeNum(void);
};

class T_CWMP_State {
private:
unsigned char ActiveKey[16];
public:           // User declarations
AnsiString __fastcall GetStateOnOff(unsigned char house, unsigned char key);
bool __fastcall GetOnOffStatus(unsigned char house, unsigned char key);
bool LightCommandMask[16][16];
bool UnitCommandMask[16][16];
bool map[16][16];
bool __fastcall LoadInfoBuff(char *buff);
};


class PACKAGE TCPU_CWMP_Socket : public TClientSocket
{
private:
    AnsiString FVersion;
    AnsiString FLogin;
    AnsiString FPassword;
    unsigned char FAuth;
    TOcelotStateChangeNum *OcelotStateChangeNum;
    T_CWMP_StateChangeNum *_CWMP_StateChangeNum;
    TOcelotState *OcelotState;

    // перевірка правильності уведення даних
    void __fastcall SetLogin(AnsiString str);
    void __fastcall SetPassword(AnsiString str);
    void __fastcall SetVersion(AnsiString)
        {
        FVersion=(AnsiString)MajorVersion+"."+(AnsiString)MinorVersion;
        }
```

```
    void __fastcall MyOnError(System::TObject* Sender, TCustomWinSocket* Socket,
TErrorEvent ErrorEvent, int &ErrorCode);
    void __fastcall MyOnConnect(System::TObject* Sender, TCustomWinSocket*
Socket);
    void __fastcall MyOnRead(System::TObject* Sender, TCustomWinSocket* Socket);
    void __fastcall MyOnDisconnect(System::TObject* Sender, TCustomWinSocket*
Socket);
    bool __fastcall SendBuf(char *buf,long size);
    // Покажчики на мої власні події
    TOnReadOcelotStatusData FOnReadOcelotStatusData;
    TOnRead10StatusData FOnRead10StatusData;
    TOnReadOtherData FOnReadOtherData;
    TOnChangeAuthStatus FOnChangeAuthStatus;
    TOnMyError FOnMyError;

    TDateTime UShortToTime(unsigned short value);
    unsigned short TimeToUShort(TDateTime time);
protected:

public:
    // конструктор і деструктор класу
    __fastcall TCPU_CWMP_Socket(TComponent* Owner);
    __fastcall ~TCPU_CWMP_Socket(void);
    T_CWMP_State *_CWMP_State;
    // запит на одержання даних про статуси
    void __fastcall SendOcelotStatusQuery(void);
    void __fastcall Send10StatusQuery(void);
    // Перенос методів з модуля
    unsigned short __fastcall GetDataFromUnit(unsigned char UnitNumber);
    unsigned char __fastcall GetIO(unsigned char UnitNumber,unsigned char
Point);
    unsigned char __fastcall GetUnitVer(unsigned char UnitNumber);
    unsigned char __fastcall GetUnitType(unsigned char UnitNumber);
    unsigned char __fastcall GetUnitsCount(void);
    TDateTime __fastcall GetCPUXADateTime();
    unsigned short __fastcall GetVariable(unsigned char VarNumber);
    TDateTime GetVarAsTime(unsigned char VarNumber);
    TDateTime GetTimerAsTime(unsigned char VarNumber);
    void SetTimeAsVar(unsigned char VarNumber, TDateTime time);
    void SetTimeAsTimer(unsigned char VarNumber, TDateTime time);
    unsigned short __fastcall GetTimer(unsigned char TimerNumber);
    // Sending commands
    void __fastcall SetIO(unsigned char UnitNumber,unsigned char Point,unsigned
char Stat);
    void __fastcall SetDateTime(unsigned char day,unsigned char mon,unsigned
char year,unsigned char hour,unsigned char min);
    void __fastcall SetVariable(unsigned char VarNumber,unsigned short Value);
    void __fastcall SetTimer(unsigned char TimerNumber,unsigned short Value);
    void __fastcall SendLeviton10(unsigned char house,unsigned char key,
unsigned char dim);
    void __fastcall SendIr(unsigned short Value);
    void __fastcall Send10Buff(unsigned char house,unsigned char key, unsigned
char repeat);
    void __fastcall Send10Command(unsigned char house,unsigned char key,
unsigned char CommandNum, unsigned char repeat);
    // Функції додаткових даних, що відносяться до програми
    void __fastcall GetInternalProtocolVersion(void);
__published:
    __property unsigned char AuthStatus = {read=FAuth};
    __property AnsiString Login = {read=FLogin, write = SetLogin};
    __property AnsiString Password = {read=FPassword, write = SetPassword};
    __property AnsiString Version = {read=FVersion, write = SetVersion};
    // мої власні події

    __property TOnReadOcelotStatusData OnReadOcelotStatusData = {read =
FOnReadOcelotStatusData, write = FOnReadOcelotStatusData};
    __property TOnRead10StatusData OnRead10StatusData = {read =
FOnRead10StatusData, write = FOnRead10StatusData};
```

```
    __property TOnReadOtherData OnReadOtherData = {read = FOnReadOtherData,
write = FOnReadOtherData};
    __property TOnChangeAuthStatus OnChangeAuthStatus = {read =
FOnChangeAuthStatus, write = FOnChangeAuthStatus};
    __property TOnMyError OnMyError = {read = FOnMyError, write = FOnMyError};

};
//------- -------------------

#endif
```

Файл about.cpp – довідка про програму

```
#include <vcl.h>
#pragma hdrstop

#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_about *Form_about;
//---------------------------------------------------------------------------
__fastcall TForm_about::TForm_about(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_about::Button1Click(TObject *Sender)
{
Form_about->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm_about::FormCreate(TObject *Sender)
{
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("МАГІСТЕРСЬКА РОБОТА");
Memo1->Lines->Add("");
Memo1->Lines->Add("на  тему:");
Memo1->Lines->Add("");
Memo1->Lines->Add("Дослідження та програмна реалізація системи платформи
управління мережними ");
Memo1->Lines->Add("пристроями інтелектуального будинку за технологією CWMP");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("Керівник: Кислун О.А.");
Memo1->Lines->Add("");
Memo1->Lines->Add("Розробив: студент Іщенко Денис Анатолійович");
Memo1->Lines->Add("                    гр. КІ-22М-2");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("м. Кропивницький 2023");
}
```

Файл about.h – бібліотека для файлу about.cpp

```
//---------------------------------------------------------------------------

#ifndef aboutH
#define aboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <jpeg.hpp>
//---------------------------------------------------------------------------
class TForm_about : public TForm
{
__published:      // IDE-managed Components
        TImage *Image1;
        TMemo *Memo1;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm_about(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_about *Form_about;
//---------------------------------------------------------------------------
#endif
```