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 першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Царенко Є.А.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Царенку Єгору Артуровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 13-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Царенко Є.А.
(прізвище та ініціали)

АНОТАЦІЯ

Царенко Є.А. Програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

Метою розробки є програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

Результат роботи – програмна реалізація системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

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: кібербезпека, захищене управління будинком

ABSTRACT

Tsarenko E.A. Cybersecurity system software for secure home management based on LanDrive technology. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system for secure home management based on LanDrive technology.

The goal of development is cyber security system software for secure home management based on LanDrive technology.

The result of the work is the software implementation of a cyber security system for secure home management based on LanDrive 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: cyber security, secure home management

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
EIB	–	європейська інсталяційна шина
X10	–	технологія інтелектуального дому

ВСТУП

Актуальність теми. Поняття "інтелектуальний будинок" було сформульовано в 70-і роки минулого століття: "Будинок забезпечуючий продуктивне й ефективне використання робочого простору". Варто розділяти поняття "інтелектуальний будинок" і "системи життєзабезпечення". Окремі системи мають лише необхідні інтерфейси керування й контролю. Концепція “Системи інтелектуального керування будинком” припускає новий підхід в організації життєзабезпечення будинку, при якому за рахунок комплексу програмно-апаратних засобів значно зростає ефективність функціонування й надійність керування всіх систем експлуатації й виконавчих пристроїв будинку.

Основною особливістю інтелектуального будинку є об'єднання окремих підсистем різних виробників у єдиний керований комплекс.

Під "інтелектуальним будинком" невірно розуміти прямий переклад з англійського як "мислячий будинок". Коректний переклад терміна intelligent building означає систему, що повинна вміти розпізнавати конкретні ситуації, що відбуваються в будинку, і відповідним чином на них реагувати: одна із систем може управляти поводженням інших по заздалегідь вироблених алгоритмах. Англійське слово intelligent, буквально означаюче "розумний", "тямущий", у сполученні зі словом building використано в значенні "гнучкий, що пристосовується". Будинок проектують таким чином, щоб всі системи його керування могли інтегруватися один з одним з мінімальними витратами, а їхнє обслуговування було б організоване оптимальним образом. Проект обов'язково припускає можливість нарощувати й видозмінювати конфігурації інстальованих систем.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

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

– Огляд існуючих систем для захищеного управління будинком на основі технології LanDrive.

– Дослідження системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

– Програмна реалізація системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захищеного управління будинком на основі технології LanDrive.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Необхідно, щоб у замовника й підрядника було сформовано єдине розуміння того, які системи й підсистеми будуть установлені. Дуже добре, якщо на той час готові проекти систем опалення, вентиляції, існує принципова електрична схема об'єкта. У цьому випадку компанія-інтегратор "інтелектуального будинку" з більшою точністю зможе додержуватися побажань замовника, які вже виражені в проектах, і спільна робота принесе більший позитивний ефект.

Далі робиться вибір слабкострумових систем, які будуть установлені в будинку: протипожежна сигналізація, системи відеоспостереження, контролю доступу, аудіо– і відео-компонента (їхнє застосування частіше характерно для житлових будинків), автоматика життєзабезпечення (керування світлом, вентиляцією й кондиціюванням, водопостачанням і опаленням, шторами та ін.), потім здійснюється підбор устаткування й надалі – проектування.

Коли відоме встаткування, спроектована провідна частина, інтегратор, генпідрядник і замовник спільно продумують сценарій роботи інтелектуальної системи, її взаємодія з користувачем – як відреагує система на натискання тих або інших клавіш.

Інтеграційні функції встаткування не повинні робити впливу на саму систему, кожний її елемент повинен мати локальну логіку. На ділі штучному інтелекту приділяється роль робити моніторинг і управляти параметрами системи на відстані в ручному або програмувальному автоматичному режимі.

У випадку збою програмного забезпечення або відмови датчиків завжди повинна залишатися можливість переходу на автономне керування кожним елементом системи.

Ще одна важлива деталь – устаткування всіх груп інтелектуальної системи повинне відповідати стандартам, прийнятим на території розміщення.

Все залежить від мети, визначеної замовником. Якщо замовник віддає переваги якомусь конкретному виробникові, він швидше за все звернеться до компанії-інтегратору, що одночасно є його дистриб'ютором. Якщо перед замовником є завдання оптимізувати витрати й при цьому одержати якісну й функціональну систему "інтелектуального будинку", його вибір концентрується на незалежній компанії-інтеграторі, не обтяженої просуванням конкретного бренду.

Оскільки ринок "інтелектуальних будинків" молодий, компанія-інтегратор часто виконує функції інсталятора встаткування. Іноді вибір інтегратора здійснюється за рекомендацією генерального підрядника будівництва. Цивілізовані тендери – їх організують великі компанії, у яких участь інтеграторів інтелектуальних систем оплачується, – проводяться рідко.

Які системи й підсистеми будуть установлені на об'єкті, як вони будуть працювати, який буде алгоритм керування ними – на ці й ще десятки інших питань дає відповідь технічне завдання або завдання на проектування. Цей багатосторінковий документ – основа всього проекту: якщо щось відсутній у технічному завданні, виходить, відсутній у проекті. Змінити проект можна, лише змінивши технічне завдання. Завдання на проектування народжується в співавторстві замовника й інтегратора. Один із ключових моментів – чітко сформульоване призначення системного комплексу "інтелектуального будинку.

Призначення Комплексу:

– дистанційний контроль / керування роботою встаткування інженерних систем;

– побудова єдиного середовища обміну даними систем контролю й керування;

– одержання оперативної інформації про стан і параметри встаткування інженерних систем;

– організація автоматизованого технічного обліку енергоресурсів;

– забезпечення оперативної взаємодії експлуатаційних служб, планування – проведення профілактичних і ремонтних робіт інженерних систем;

– підвищення надійності, безпеки, і якості функціонування встаткування інженерних систем;

– реєстрація й створення архіву технологічних процесів інженерних систем і дій експлуатаційних служб;

– скорочення експлуатаційних витрат;

– зниження загальної кількості встаткування за рахунок уніфікації й підвищення повноти – використання функціональних можливостей устаткування одночасно в різних підсистемах.

З боку замовника обов'язковим є надання "вихідних даних для проектування":

– поетажні плани з експлікацією приміщень;

– робочі проекти інженерних систем, виконані розроблювачами суміжних розділів і об’єкти автоматизації: опалення, холодне й гаряче водопостачання, вентиляція й кондиціювання, каналізація, центральний тепловий пункт, індивідуальний тепловий пункт, електроустаткування й електроосвітлення з пояснювальними записками й характеристиками встаткування;

– вихідні дані по позаплощадкових і внутрімайданчикових інженерних мережах.

У завданні обов'язково повинні бути присутнім всі інженерні системи, що підлягають розробці й впровадженню в рамках створюваного проекту "інтелектуального будинку". Наприклад:

– автоматична пожежна сигналізація;

– автоматичне пожежогасіння;

– системи охоронної сигналізація, охоронного відеоспостереження, керування й контролю доступу;

– системи зв'язку й телекомунікацій.

Також у технічне завдання необхідно включити короткий опис комплексу автоматизації й диспетчеризації систем будинку, що повинен забезпечити автоматичне керування, регулювання, необхідні блокування, захисти від аварійних режимів, індикацію сигналів тривоги й порушення режимів регулювання, оптимізацію роботи інженерного встаткування, економію енергоресурсів, централізований автоматичний контроль і облік витрат:

1.2 Область застосування

Автоматизації й диспетчеризації підлягають наступні системи:

1. Механічні системи – холодопостачання:

– загальобмінна вентиляція;

– кондиціювання;

– підпір повітря й противодимна вентиляція;

– холодне водопостачання;

– гаряче водопостачання;

– побутова каналізація;

– протипожежний водопровід;

– центральний і індивідуальні теплові пункти;

– опалення.

2. Електричні системи – розподільна електромережа до поверхових розподільних щитів:

– робоче освітлення під'їздів, сходових кліток;

– аварійне освітлення під'їздів, сходових кліток;

– евакуаційне освітлення;

– зовнішнє освітлення.

3. Системи безпеки:

– автоматична пожежна сигналізація;

– голосове оповіщення про пожежу;

– автоматичне пожежогасіння;

– охоронна сигналізація;

– охоронне відеоспостереження;

– контроль доступу.

4. Диспетчеризація ліфтів:

– аварійний спуск на 1-й поверх.

5. Системи зв'язку й телебачення:

– телефонний зв'язок;

– радіотрансляція;

– супутниковий телеприйом;

– часофікація;

– локальна обчислювальна мережа;

– високошвидкісний вихід в Internet.

6. Системи обліку й керування споживанням наступних ресурсів:

– електроенергії (на введенні, загальбудинкові витрати й по споживачах);

– холодної води (на введеннях, загальбудинкові витрати й по споживачах);

– гарячої води (загальдомовики витрати й по споживачах);

– опалення (роздільно житло й комерційні площі);

– холодопостачання (по споживачах).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Розумний Будинок MimiSmart

Керування освітленням. Життя в будинку неможливо без використання світла. А в сучасних будинках і квартирах лампочок, люстр, світильників, підсвічувань та ін. безліч. І природне бажання людини – мати можливість легко управляти цим царством світла. Для цього й існує система керування освітленням «Розумний Будинок», що дозволяє управляти освітлювальними приладами, не додаючи майже ніяких зусиль.

Керування електроприладами. Тепер це можна робити легко, навіть сидячи в зручному кріслі. Просто скористайтеся єдиною панеллю керування системи «Розумний Будинок», настроївши всі прилади на потрібний режим роботи, або контролюйте їх через мобільний телефон.

Керування кліматом. За цим ретельно стежить система керування кліматом «Розумний Будинок». Усе, що потрібно, – це внести необхідні настроювання для кожної кімнати, і можна насолоджуватися сприятливим кліматом у будинку цілий рік.

Керування вентиляцією. Щоб у будинку був здоровий мікроклімат, необхідна надійна система вентиляції, вентиляція з інтелектом. Така, як в "розумному" будинку.

Розумний Домофон. Домофон в «розумному» будинку – це не просто кнопка виклику, що відкриває й закриває двері. Це ціла система з набором життєво важливих для комфортного життя функцій.

Мультирум. Сьогодні ми не уявляємо собі життя без телевізора, комп'ютера, музичного центра й іншої електронної техніки, який від голови до ніг напхані сучасні житла. Погодитеся, громіздко й некрасиво. Але тільки не в «розумному» будинку, адже він обладнаний системою мультирум, що з одного джерела розподіляє відео– і аудіосигнали по всіх кімнатах житла. Перебуваючи в будь-якій частині будинку, ви можете слухати музику або дивитися відео, навіть якщо джерело сигналу перебуває зовсім в іншій кімнаті.

Керування системою поливу. Важливо зберегти красу саду й кімнатних квітів. Про це й піклується система автоматичного поливу «Розумний Будинок». Вона, як уважний садівник, робить все точно в строк і з особливою акуратністю.

Статистика. Контроль витрат, постійний з'єм показання лічильників тепла, води, електроенергії, щоб ураховувала споживання ресурсів і щомісяця без запізнень оплачуваня рахунку.

Керування Розумним будинком. Завдяки сучасній сенсорній панелі, керування «Розумним будинком» – це не просто повсякденне завдання, це найцікавіший захоплюючий процес, що доставляє максимум комфорту й задоволення, і він настільки простій, що його швидко освоїть навіть дитина.

Відеоспостереження. У «розумному» будинку встановлена інтелектуальна система відеоспостереження.

Безпека. «Розумний» будинок оснащений інтелектуальною системою охорони.

Система керування будинком ТОВ "Будівельна компанія "Київбудрезерв"

Відомо, що сучасне заміське житло – буквально нашпиговане різного роду інженерними системами. Це й численні освітлювальні прилади, системи газо– і водопостачання, опалення, клімат-контролю, інформаційні комунікації: телефон, Інтернет, супутникове й кабельне телебачення. Є й системи забезпечення безпеки й охорони хазяїв (від зловмисників і від пожежі). Не обходиться будинок і без домашнього кінотеатру й інших зручностей, покликаних скрасити відпочинок.

Все це – пристрої, роботою яких необхідно управляти. Традиційний підхід до рішення проблеми має на увазі наявність безлічі всіляких вмикачів, перемикачів, кранів, вентилів і пультів дистанційного керування. «Розумний будинок» змушує всі системи й прилади взаємодіяти відповідно до різних сценаріїв. Сценарії задаються за допомогою домашнього комп'ютера або убудованого процесора.

Ви зможете управляти будь-яким устаткуванням у будинку з одного універсального пульта. А якщо пульт раптом зламається, тимчасово прийде користуватися звичайними вимикачами або сенсорними панелями. Більше того, керувати «розумним будинком» пропонується й дистанційно – через Інтернет, або за допомогою мобільного телефону. Наприклад, виїжджаючи з міста, «замовити» собі сауну або теплу воду в басейні.

Всі завдання, пов'язані з керуванням домашньою технікою, можна перекласти на мікропроцесорний контролер (або кілька контролерів).

Рисунок 2.1 – Інтерфейс користувача системи керування будинком ТОВ "Будівельна компанія "КиївБудРезерв"

Керування освітлювальними приладами. Спеціальні пристрої плавно регулюють освітленість вашого житла в різних кімнатах. Вони ж регулюють розжарення ламп, що істотно заощаджує електроенергію. При бажанні можна не тільки управляти рівнем освітленості різних кімнат, але й створити особливе світлове оформлення усередині однієї з них, залежно від ситуації – романтичне побачення, ділова зустріч або сімейне торжество. Таке чутливе освітлення встановлюють і навколо будинку. Усе більше доступними і масовими стають датчики руху. На їхній основі легко організувати автоматичний супровід світлом людини, що йде по будинку.

Системи клімату-контролю. Системи кліматичного контролю регулюють температуру й вологість у приміщенні, відповідають за роботу кондиціонерів і опалювальних приладів. Мікроклімат буде підтримуватися по заданій вами програмі. Ви також можете управляти температурою в будинку за допомогою кімнатного терморегулятора.

Керування охоронною й пожежною сигналізацією. Організація охорони вашого будинку забезпечується різними способами. Часто використовують оповіщувачі, що сигналізують про проникнення нежданих гостей. Вони включають системи спеціального оповіщення – спрацьовує сигнальна миготлива лампа, сирена, надходить дзвінок у міліцію й т.д., усе, що ви запрограмуєте для цього випадку. Завдання захисту від зломів і крадіжок вирішується насамперед за рахунок добре продуманої системи сигналізації. В «розумному будинку» дуже часто використовуються два контури сигналізації: зовнішній, до якого підключені всі датчики відкриття вікон і дверей по периметрі будинку, і внутрішній, що складається з датчиків руху, розташованих усередині. Ідучи на роботу, ви включаєте обидва контури. А, наприклад, увечері можна задіяти тільки зовнішній контур і спокійно відпочивати з родиною, зовсім не турбуючись про те, що місцеві хулігани проникнуть у будинок через вікно комори.

Розумний будинок може бути обладнаний і системою відеоспостереження. Вона буде невпинно стежити за що відбувається як усередині, так і зовні. Відомо,

що вдома частіше піддаються злому під час відсутності хазяїв. Тому, їдучи, ви можете дати системі завдання створювати ефект присутності хазяїв. У кімнатах буде включатися й вимикатися світло, жалюзі на вікнах стануть відкриватися й закриватися. Навіть найближчі сусіди не догадаються, що в будинку нікого немає. Ще одна зручна функція, що реалізується в усіх без винятку системах «розумний будинок», умовно називається «Виключити все». Ідучи останнім, ви натискаєте кнопку «Нікого немає будинку», і у всьому будинку відключаються світильники й всі розетки, у яких, можливо, залишилася включена праска. Одночасно перекривається трубоподача газу, холодної й гарячої води, система опалення починає працювати в економічному режимі, зовнішній і внутрішній контури системи сигналізації ставляться на охорону.

Smarthome Manager Essential

Smarthome Manager Essential працює під керуванням операційних систем Microsoft Windows.

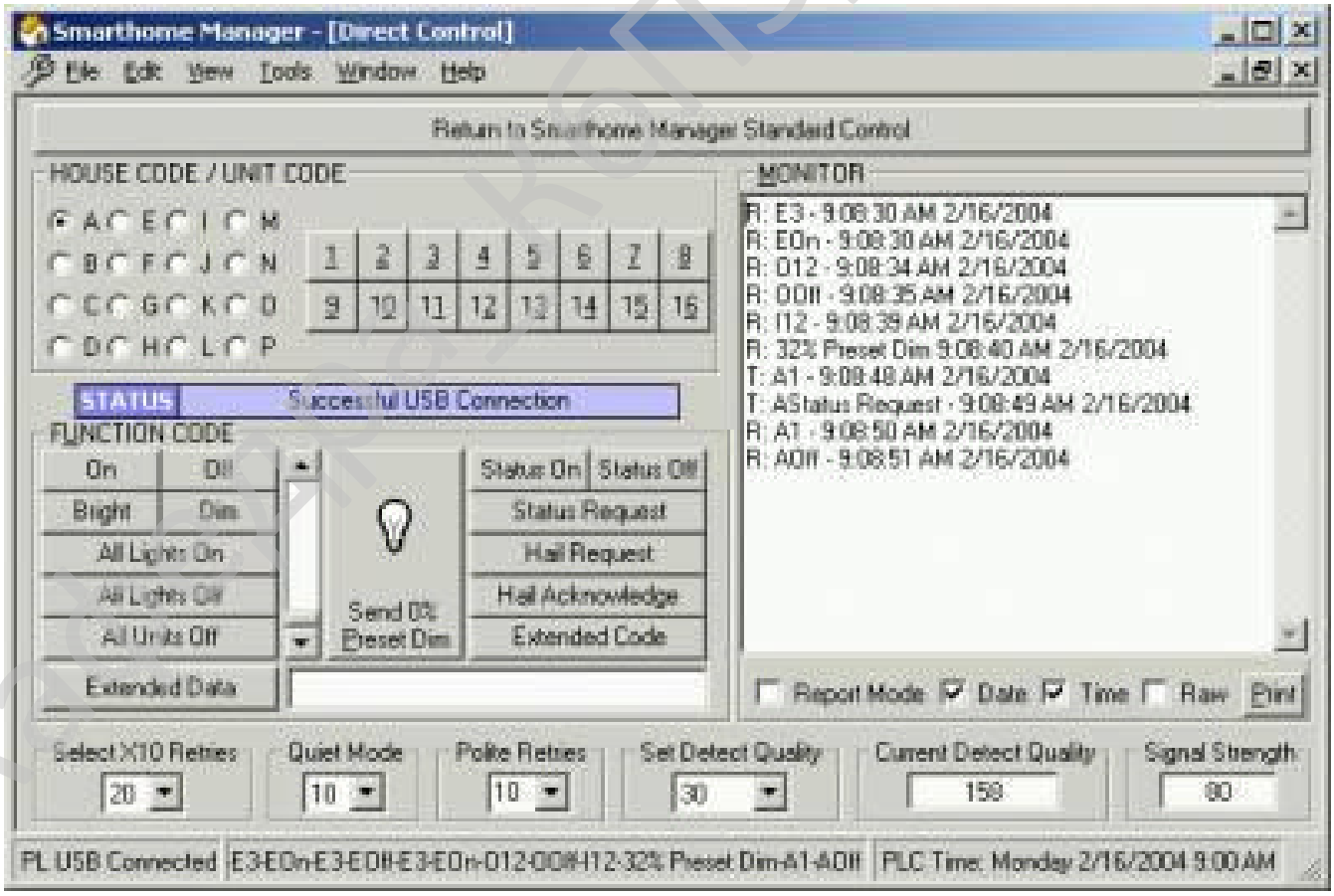

Рисунок 2.2 – Інтерфейс користувача Smarthome Manager

Ця програма поставляється в комплекті з Smarthome PowerLinc Controller USB і іншими X 10-сумісними комп'ютерними контролерами компанії Smarthome. Повну версію Smarthome Manager можна встановити в будь-який час або вам, можливо, удасться купити набір, що включає в себе PowerLinc Controller USB і Smarthome Manager.

Повна версія Smarthome Manager працює також як Essential, але вона може автоматично визначати деякі типи модулів і створювати додаткові умови усередині дій. У цьому розділі за назвою Smarthome Manager будуть матися на увазі обидві версії за винятком тих випадків, коли будуть розглядатися можливості, наявні тільки в повної версії. Після запуску Smarthome Manager відкривається вікно з інтерфейсом, схожим на Windows Explorer (Провідник) із чотирма об'єктами:

– Locations (Місце розташування).

– All Devices (Всі пристрої).

– Event Triggered Action (Дії, ініційовані подіями).

– Time Triggered Action (Дії, виконувані за часом).

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних

багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++

використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки 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 кібербезпеки 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

Сучасне житло людини, будь то котедж або міська квартира, надзвичайно складний по своїй технічній оснащеності об'єкт. Він начинений світильниками, кондиціонерами, холодильниками, вентиляційним устаткуванням, аудіо-відеотехнікою, системами відеоспостереження, охоронною сигналізація й багато чим ще. Все це вимагає постійного включення-вимикання й невсипущого контролю.

От перелік пристроїв, якими потрібно управляти:

– системи життєзабезпечення (електрика, газ, водопровід, опалення, освітлення);

– інформаційні системи (телефон, Internet, супутникове й кабельне телебачення);

– пристрою забезпечення безпеки (у тому числі й пожежної) і охорони;

– аудіо-відеосистеми, домашній кінотеатр та інші компоненти культурно-дозвільного комплексу.

Зібрати в ціле й додати системі завершеність у вигляді загального пульта керування – це і є завдання побудови Системи керування будинку. Система повинна жити як єдиний організм, що реагує на зміни зовнішніх і внутрішніх факторів. Ваші обов'язки по керуванню домашньою технікою будуть перекладені на мікропроцесорний контролер (або кілька контролерів). Можливості, які дає ця система безмежні.

Керування освітлювальними приладами

Це одна із самих корисних сторін Системи керування будинку. Спеціальні пристрої плавно регулюють освітленість вашого житла в різних кімнатах. Це можна робити вручну, а можна встановити датчик рівня освітленості. Якщо за

вікнами стане темніти, то прилад подасть сигнал лампі, що негайно ввімкнеться. Якщо ж, навпроти, сонце піднялося, а ви не замовляли яскраве освітлення, то по команді датчика плавно закриються жалюзі. Вони ж регулюють розжарення ламп, що істотно заощаджує електроенергію.

При бажанні можна не тільки управляти рівнем освітленості різних кімнат, але й створювати особливе світлове оформлення усередині однієї з них, залежно від ситуації – на випадок романтичного побачення, наприклад. І не тільки усередині – при бажанні таке чутливе освітлення можна встановити й навколо будинку.

Останній писк – датчики руху. На їхній основі досить легко організувати автоматичний супровід світлом людини, що йде по будинку.

Системи клімат-контролю

Системи кліматичного контролю регулюють температуру й вологість у приміщенні, відповідають за роботу кондиціонерів і опалювальних приладів. Мікроклімат буде підтримуватися по заданій вами програмі. Ви також можете управляти температурою в будинку за допомогою кімнатного терморегулятора. Крім створення ідеальних температурних умов для життя, це також дозволяє заощадити на електриці.

Керування охороною й пожежною сигналізацією

Організація охорони вашого будинку з розумом забезпечується різними способами. По-перше, це оповіщувачі, що сигналізують про проникнення нежданих гостей. Вони включають системи спеціального оповіщення – спрацьовує сигнальна миготлива лампа, сирена, дзвінок у міліцію й усе, що ви запрограмуєте для цього випадку.

По-друге, будинок з розумом може бути постачений відеокамерою або їхньою мережею. Вони будуть невпинно стежити за що відбувається як усередині будинку, так і зовні. Зображення можна вивести на екран комп'ютерного монітора або телевізора.

Відомо, що взлому частіше піддаються будинку, хазяї яких відсутні. Тому, їдучи, ви можете дати Інтелектуальному будинку завдання створити ефект присутності в будинку на час вашого від'їзду. У кімнатах буде включатися-вимикатися світло, відкриватися-закриватися жалюзі. Навіть найближчі сусіди не догадаються, що в будинку нікого немає.

Неуважні люди, які ніколи не впевнені, чи виключили вони праска або телевізор, установивши в приміщенні димові оповіщувачі, можуть нарешті зітхнути спокійно. Якщо ці прилади "відчують" дим, то вони самостійно будуть додзвонюватися в пожежну частину, хазяїнові на роботу або на мобільний телефон.

Домашні кінотеатри

Отже, культурно-дозвільний комплекс вашого будинку: тут в основі є той же системний підхід. По-перше, вся аудіовідеотехніка зв'язана в єдину мережу. По-друге, система охоплює цілий будинок – управляти можна з будь-якого куточка. Великій родині не обов'язково встановлювати в кожній кімнаті й відеомагнітофон, і аудіосистему – досить мати одну загальну на всіх: нові кабельні мережі дозволяють передавати відео– і аудіосигнали на приймаючі в будь-якій кімнаті. Отут-то й виникає загадкове поняття "мультирум", що означає джерела аудіо-відеосигналів, зібрані в єдиній стійці, у сполученні із системою розподілу A/V даних по будинку, що живлить звуко-візуальну атмосферу в інших кімнатах.

До того ж аудіо-відеотехніка з'єднана з усіма іншими системами. Тому, коли ви збираєтеся подивитися фільм, одночасно включається домашній кінотеатр, закриваються жалюзі, освітлення плавно гасне. До останнього часу словосполучення "домашній кінотеатр" асоціювалося з більшим телевізором, оточеним аудіоапаратурою. Однак в останні роки домашні кінотеатри звичайно створюють на основі відео проекторів і більших екранів, або на базі плазмених панелей.

А взагалі всіма встановленими у вашім будинку системами можна управляти не тільки за допомогою дистанційних пультів, про які ми згадували вище. При наявності необхідної програми вони будуть озиватися на ваш голос. Із Системою керування будинком ви можете зв'язатися з будь-якої точки земної кулі через Internet або по телефоні.

До речі, не тільки ви можете впливати на систему. Інтелектуальний будинок сам може нагадувати вам про те, що необхідно зробити, голосовим способом або по телефону. При автоматизації свого будинку кожний може вибрати систему, найбільш підходящу йому по функціях, складності й вартості. Звичайно, найкраще подумати про те, що будинок повинен бути інтелектуальним, ще на етапі будівництва. Але якщо ремонт уже зроблений, а автоматизувати будинок дуже хочеться, то й це можливо. Питання тільки в тому, у яку компанію ви звернетеся і якими засобами ви розташовуєте.

Система безпеки

В "розумному будинку" система безпеки побудована таким чином, щоб якомога раніше виявити й запобігти загрозливим життям і здоров'ю людини ситуації, ліквідуючи тим самим причини наших найпоширеніших "побутових" страхів.

Захист від зломів і крадіжок. Це завдання вирішується насамперед за рахунок добре продуманої системи сигналізації. В "розумному будинку" дуже часто використовуються два контури сигналізації: зовнішній, до якого підключені всі датчики відкриття вікон і дверей по периметрі будинку, і внутрішній, що складається з датчиків руху, розташованих усередині будинку. Ідучи на роботу, ви включаєте обидва контури. А, наприклад, увечері можна задіяти тільки зовнішній контур і спокійно відпочивати з родиною, зовсім не турбуючись про те, що місцеві хулігани проникнуть у будинок через вікно комори.

Цих же самих хуліганів і інших зловмисників "розумний будинок" легко введе в оману щодо присутності хазяїв. По заданій програмі буде включатися й

вимикатися світло, відповідно піднімуться або опустяться жалюзі, періодично буде гавкати собака. Всі ці "хитрості" припиняють злочинні наміри випадкових і зовсім не випадкових перехожих. Якщо ж зловмисники все-таки захочуть проникнути у ваш "розумний будинок", їх злякають завивання сирени й миготіння світла, а на місцевий пост охорони й (або) ваш мобільний телефон відразу надійде сигнал про злом.

Захист від пожежі, витоків і витоку газу. Пожежна сигналізація в заміському будинку або квартирі – це вже норма життя. Система "розумний будинок" дозволяє реалізувати всі стандартні функції протипожежної сигналізації (наприклад, автоматичне відключення вентиляції при пожежі, звукове й світлове оповіщення). Для запобігання витоків у ванній, туалеті й на кухні встановлюються датчики, які контролюють появу води. При виявленні витоку автоматично закриваються клапани на трубопроводах, що подають, холодного й гарячого водопостачання, а на ваш стільниковий телефон або по електронній пошті надходить повідомлення про аварійну ситуацію. Точно так само система "розумний будинок" реагує й при витоку газу в котельні або на кухні.

Додаткові функції безпеки. Навіть якщо до вас в "розумний будинок" під видом електрика проникнув підозрілий суб'єкт, ви завжди зможете подати сигнал тривоги, усього лише на парі секунд довше потримавши палець на кнопці звичайного вимикача, увімкнути світло. При цьому "незваний гість" навіть не запідозрить, що ви комусь про щось повідомили.

Ще одна з дуже зручних функцій, що реалізується в усіх без винятку системах "розумний будинок", умовно називається "Виключити всі". Ідучи останнім, ви натискаєте кнопку "Нікого немає будинку", і у всьому будинку відключаються світильники й всі розетки, у яких, можливо, залишилися включеними забута праска або фен. Одночасно перекриваються трубопроводи газу, холодної й гарячої води, система опалення починає працювати в економічному режимі, зовнішній і внутрішній контури системи сигналізації ставляться на охорону.

Первісні вкладення в систему "розумний будинок" перевищують витрати на стандартну систему керування електроустаткуванням. Мається на увазі зниження витрат при експлуатації "розумного будинку" за рахунок економії енергоресурсів (холодної й гарячої води, газу, електрики). Поки в нашій країні ця економія не настільки значна, як у країнах Європи, але, по всіх прогнозах, у найближчому майбутньому ціни на теплоносіїв і електроенергію істотно виростуть. Ще одна немаловажна стаття економії випливає з реалізованих "розумним будинком" функцій безпеки. Запобігання аварійних ситуацій, у першу чергу витоків і пожеж, знижує до мінімуму ймовірність непланових ремонтів.

Далі, у даному розділі розглянемо основу технології LanDrive, на якій будуються системи управління будинками.

LanDrive – це універсальна платформа для побудови класичних шинних розподілених систем управління в системах «розумний будинок», в автоматизації будинків. LanDrive – найбільш доступна на сьогоднішній день платформа для побудови шинних розподілених систем керування внутрішнім і вуличним освітленням, силовими навантаженнями, електроприладами, а також такими системами, як опалення, кондиціювання, вентиляція, охоронна сигналізація, контроль доступу й утікань води. Також можливе керування аідио– і відеотехнікою, домашніми кінотеатрами, жалюзі, рольставнями, шторами, воротами, насосами, двигунами. В основному орієнтована на застосування в складі «розумного будинку», але останнім часом всі частіше застосовується в системах обліку й заощадження енергоресурсів, контролю доступу, охоронно-пожежних системах.

Структурно система складається із центрального контролера й виконавчих модулів, зв'язаних між собою польовою шиною (мережею). До виконавчих модулів підключаються кероване устаткування.

Для взаємодії система використовує на фізичному рівні стандарт RS-485. Для взаємодії на прикладному рівні використовується широко відомий протокол Modbus/RTU. Устаткування підтримує швидкість обміну інформацією в режимах

2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 біт/с, що, досить для миттєвого виконання команд. Максимальна кількість пристроїв, керованих одним головним контролером (ПЛК) – 100. Можливе об'єднання декількох головних контролерів в одну мережу. Через використання стандартних протоколів можливе включення в систему устаткування сторонніх виробників, також можливе використання пристроїв серії LanDrive в інших системах.

Елементи системи:

– Центральний керуючий контролер, що дозволяє управляти всіма пристроями за заданими сценаріями.

– Релейні модулі, що забезпечують комутацію потужного навантаження до 1,2 кВт, а також опитування датчика типу «сухий контакт».

– Діммируючі модулі, що забезпечують можливість плавної зміни потужності в навантаженні.

– Модулі інфрачервоного зв'язку, що забезпечують прийом команд із пультів дистанційного керування й передачу збережених команд на побутові пристрої.

– Модулі з 4-мя цифровими входами, що забезпечують опитування декількох датчиків.

– Модулі розширення з аналоговими входами-виходами, що забезпечують опитування аналогових датчиків, наприклад, температури, і керування аналоговими пристроями.

– Модулі й датчики збору інформації: температури, вологості, руху.

– Опціональні комунікаційні модулі GSM.

– Блоки живлення всієї системи й центрального контролера.

У цій системі необхідне програмування тільки головного контролера. Для програмування використовується спеціалізоване програмне забезпечення – LanDrive Configurator Pro.

Система може управлятися за допомогою великого переліку програмного забезпечення:

– SCADA-додатки.

– OPC-сервери Modbus.

– Спеціальні додатки для керування будинками, розумними будинками, такі як IridiumMobile, IdomMedia.

Весь перелік додатків дозволяє здійснювати оперативне візуалізоване керування розумним будинком, у тому числі:

– керування системою LanDrive із сенсорних моніторів, панелей;

– керування системою LanDrive з кишенькових комп'ютерів;

– керування системою LanDrive з iPhone, iPod, iPad;

– керування системою LanDrive через Інтернет по захищеному каналу;

– об'єднання безлічі центральних контролерів SPIDER у єдину мережу;

– створення власних віртуальних панелей керування будинком;

– можливість інтеграції з устаткуванням сторонніх виробників.

Центральний контролер

Центральний контролер може використовуватися як у настільному варіанті, так і монтуватися на DIN-рейку. Програмується з персонального комп'ютера за допомогою програми LanDrive Configurator через USB-порт або Ethernet-підключення. Після програмування комп'ютер не використовується для роботи системи. Центральний контролер є високоінтегрованим пристроєм у широким набором периферії. За допомогою нього можливе створення:

– дротових мереж "розумного" будинку;

– бездротових мереж "розумного будинку";

– комплексні системи збору й обліку даних про споживання ресурсів, АСКУЕ по Ethernet, Internet

– систем охоронно-пожежної сигналізації, у тому числі й по мережах GSM;

– систем оповіщення й вилученого керування по мережах GSM, Ethernet, Internet

– систем доступу;

– систем витоку води й витоку газу;

– систем клімат-контролю;

– систем керування встаткуванням "мультирум";

– систем диспетчеризації, розподіленого керування й моніторингу по інтернет і локальні мережі, у тому числі груп будинків у котеджних селищах, стільникових станцій, очисних споруджень, серверних по мережах GSM, Ethernet, Internet

– систем автоматичного введення резерву (АВР);

– систем керування котельнями по мережах GSM, Ethernet, Internet.

Релейні модулі

Керовані релейні модулі випускаються у двох виконаннях – мініатюрному для вбудовування, наприклад, у стакани розеток під вимикачі й розетки, і для установки на DIN рейку. Пристрої, установлювані на DIN-рейку, мають більшу потужність.

Діммуючі модулії

Керовані діммуючі модулі випускаються у двох модифікаціях – потребуючі підведення нуля й фази, і "розривні". Другі зручні для монтажу в склянки вимикачів, де не проведений нуль.

Модулі інфрачервоного зв'язку

Трансівери (приймачепередатчики) інфрачервоних сигналів виконуються у вигляді, що вбудовуються. Комплектуються шліфованою металевою панеллю. Є універсальними, дозволяють розпізнавати, запам'ятовувати й відтворювати ІЧ-команди будь-якого виробника.

Модулі із цифровими входами

Випускаються в мініатюрних корпусах, що дозволяє вбудовувати в стандартні стакани вимикачів. До клем модулів підключаються датчики виходи, що мають, типу "сухий контакт".

Модулі розширення

Розширені модулі для підключення датчиків з універсальною кількістю входів і виходів керування. Випускаються в DIN-корпусах.

Датчики

Датчики руху, освітленості й т.д. Можливо підключення датчиків різних виробників.

Блоки живлення

Стабілізовані промислові блоки живлення. Система на вибір комплектується блоком живлення відповідної потужності.

Монтаж

Монтаж системи ведеться класичною екранованою крученою парою FTP. Кабель приховано прокладається по периметру приміщення з відгалуженнями до кожного пристрою LanDrive. У місці з'єднання, у кабелю знімається зовнішня й внутрішня оплітка на відрізку приблизно 1,5-2,0 см., далі він скручується й закріплюється обтиск наконечниками. Якщо монтаж здійснюється раніше установки модулів на тривалий час, необхідно не розрізаючи шину, залишити запаси кабелю (петлі) у місцях майбутнього розташування модулів. Кабель підводить до всіх стаканів вимикачів, розеток (якщо необхідно їхню автоматизацію), в усі стелі. Крім цього необхідно прокинути кабель по всіх периметрах всіх приміщень. Це вирішить всі питання по раптовій зміні планів точок автоматизації замовниками.

3.2 Розробка структурної схеми

Опишемо основні складові інтелектуального будинку.

Інформаційні системи:

– локальна комп'ютерна мережа;

– домашній офіс;

– широкополосний доступ у глобальну мережу (Інтернет).

Безпека:

– пожежна сигналізація й система автоматичного пожежогасіння;

– охоронна сигналізація;

– система контролю доступу;

– система зовнішнього й внутрішнього відео спостереження;

– система управління паркінгом;

– система внутрішнього оповіщення (радіомережа);

– аварійний контроль інженерних систем;

– система екологічного контролю.

Зв'язок:

– телефонний зв'язок внутрішня (інтерком) і локальні АТМ;

– телефонний зв'язок зовнішня провідна, радіорелейна, супутникова;

– Інтернет-телефонія;

– системи відеоконференцій.

Мультимедіа:

– наземне й кабельне телебачення;

– супутникове телебачення;

– домашнє відео;

– домашній кінотеатр;

– мульти– аудіо й відео (multiroom).

– один пульт управління для всіх систем

Система електроживлення:

– безперебійне й гарантоване електропостачання;

– захист від поразки електричним струмом людей і тварин;

– автоматизація управління електроживленням побутових приладів;

– запобігання перевантажень і короткого замикання в електричній мережі;

– управління якістю електроживлення – моніторинг, стабілізація й фільтрація;

– облік витрат й управління споживанням електроенергії.

Освітлення внутрішнє й зовнішнє:

– аварійне й чергове освітлення;

– автоматизація управління світлом;

– гнучке настроювання світлових груп;

– дистанційне й віддалене управління світлом;

– програмування світлових сцен.

Опалення, вентиляція, кондиціювання:

– автоматизація управління температурою повітря;

– контроль якості повітря;

– узгодження роботи різних кліматичних систем;

– облік витрати й управління споживанням теплової енергії.

Водопостачання й газопостачання:

– автоматичне наповнення ванн, басейнів і накопичувальних резервуарів;

– автономне й резервне нагрівання води;

– запобігання витоку водопроводу й витоку газу;

– управління ландшафтними водяними системами (фонтани, водоспади);

– управління температурою води у ваннах і басейнах (термостатування);

– облік витрат й управління споживанням води й/або газу.

Тепер спробуємо створити зі звичайного будинку інтелектуальний. Для цього нам потрібно всі прилади, які виконують перераховані вище функції, об'єднати в одну систему й підключити її до віддаленого сервера. Тим самим буде управління кожним компонентом, та всіма воєдино. І необхідно щоб контроль за станом всіх приладів користувач міг одержати в будь-який момент часу, навіть перебуваючи поза будинком. Для цього треба організувати мережу в будинку, об'єднавши всі прилади й системи над якими треба здійснити контроль, та потім вибрати спосіб підключення до віддаленого сервера.

На рисунку 3.1 зображена узагальнена структурна схема системи управління домом.

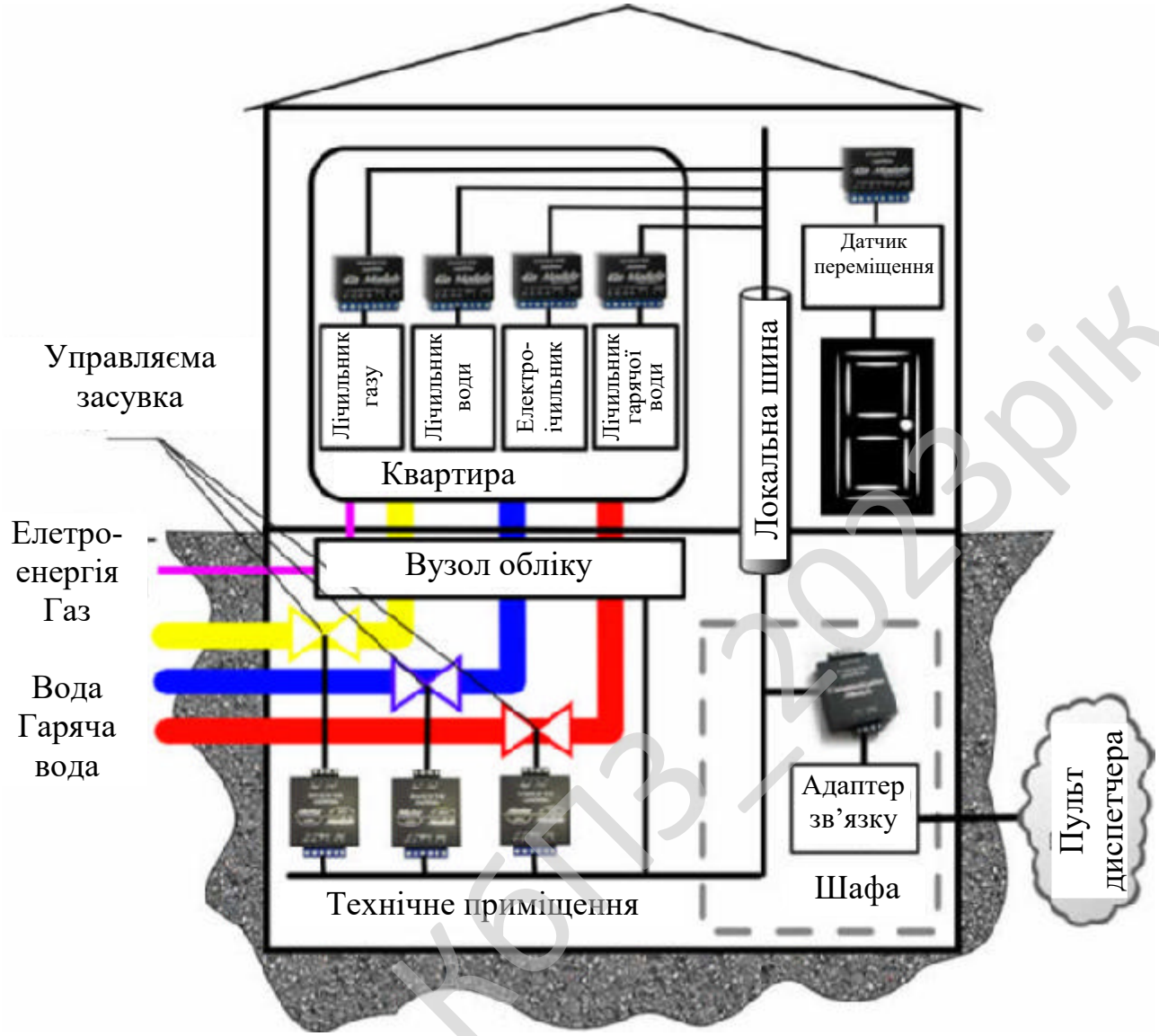

Рисунок 3.1 – Структурна схема системи

Система складається з набору модулів LanDrive, об'єднаних у єдину мережу по інтерфейсу RS-485 на основі крученої пари. Всі лічильники, датчики й виконавчі механізми підключаються до модулів системи LanDrive по інтерфейсах типу сухий контакт, рахунковий і аналоговий входи, а також через релейні виходи модулів.

Система заживляється постійною напругою 12-24В силою току з розрахунку 0,5Вт на один модуль. Для заживлювання системи передбачається резервне джерело живлення, бажано автономне на 24 години функціонування.

Як середовище передачі даних на диспетчерський пульт використовується мережа провайдеру з виділеним VPN-каналом. Фізичне підключення здійснюється за допомогою Ethernet-адаптера с виділеним IP-адресою. Для передачі використовується шифрування SSL 128 біт.

Для взаємодії на прикладному рівні використовується широко відомий протокол Modbus/RTU.

Як було відзначено вище, сучасний інтелектуальний будинок – це дуже складне інженерне рішення, що складається з наступного набору систем:

– Система безпеки.

– Система комфорту.

– Інформаційна система.

– Система диспетчеризації.

Розглянемо які підсистеми входять у перераховані вище системи.

Система безпеки:

– Система цифрового відеоспостереження з можливістю одночасного спостереження, перегляду, архівування. Режим віддаленого перегляду й управління через Інтернет.

– Бездротова пожежна й охоронна сигналізація з можливістю обміну інформацією через GSM модуль.

– Система контролю доступу в приміщення (у тому числі віддалене управління гаражними воротами).

Система комфорту:

– Внутрішня телефонна система, голосний зв'язок усередині будинку.

– Система супутникового, ефірного телебачення з можливістю перегляду в будь-якій кімнаті.

– Система "Домашній кінотеатр".

– Система "Мультирум" аудіо– й відео– (система "звук навколо").

– Цифровий розважальний комплекс, оцифровка відео, печать фотографій, створення особистих цифрових фото– і відео– альбомів і т.д.

– Управління світлом у всьому будинку, світлові сцени й сценарії.

– Управління системою вентиляції й кондиціювання.

– Управління системою опалення.

– Управління сауною, басейном.

Інформаційна система:

– Установка локальної обчислювальної мережі в будинку, мережна печать, мережні ігри (можлива використання бездротових технологій).

– Вихід в Інтернет з будь-якого комп'ютера в будинку (у тому числі з мобільного).

– Віддалене управління всіма системами будинку через Інтернет.

– "Домашній офіс" з віддаленим підключенням до корпоративної мережі робочого офісу.

Система диспетчеризації:

– Система безперебійного електропостачання

– Управління системою опалення, казаном водонагрівача

– Контроль витоку води, газу.

– Система управління здатна погоджувати роботу інженерних систем, оцінюючи стан сенсорів, датчиків, відпрацьовуючи команди з пультів управління, прив’язуючись до часу доби, пори року й т.п. При цьому виключаються ситуації, коли домашнє устаткування, покликане вирішувати спеціалізовані проблеми, працює в режимах, взаємовиключаючих один одного.

Інтелектуальний будинок дозволяє замінити всі пульти управління однією або декількома (по кількості зон або кімнат) сенсорними панелями. Вони дозволяють не ламати голову над тим, як підбудувати середовище перебування під необхідні умови.

Використання сучасного устаткування дозволяє створити в будинку єдиний комплекс із систем безпеки (охоронна й пожежна сигналізація, відеоспостереження, контроль доступу), систем зв'язку й комунікації (телефонна

й комп'ютерна мережі, оповіщення, екстрений виклик), систем управління опаленням, вентиляцією, освітленням і т.д., що працює по обраному алгоритму.

Як ілюстрацію можна привести приклад, коли автоматизація й включення в єдиний контур управління освітлювальної системи й систем клімат-контролю (опалення, кондиціювання й вентиляція) будинку (окремої квартири) дозволяють реалізувати автоматичне управління цими системами залежно від пори року й доби, умов навколишнього середовища, присутності людей і інших ,факторів. У результаті досягається істотне зниження витрат на електроенергію й теплопостачання. Досвід показує, що економія експлуатаційних витрат у цьому випадку може досягати 15-20%.

Состав системи:

– Центральний процесор. Обробка керуючих команд від керуючих сенсорних панелей, відправлення команд для ІЧ банків, диммерів, релейних модулів. Обробка вхідної информації від датчиків, контролерів, модемів. Установлюється програмне забезпечення розроблене у результаті виконання бакалаврського проектування, його серверна частина.

– Керуючі сенсорний панелі. Передача в центральний процесор команд керування, відеомоніторинг, подання на екрані всієї необхідної інформації/пов'язані із центральним процесором по мережі Ethernet (TCP/IP)

– Діммери. Регулювання яскравості світіння ламп, управляються центральним процесором або по командах від сенсорних панелей або відповідно до закладеного алгоритму (сценарію), пов'язані із центральним процесором RS-485 інтерфейсом.

– Релейні модулі. Керування релейними контактами й прийом сигналів на входи типу 0-1 управляються Центральним процесором або по командах від Керуючих сенсорних панелей або відповідно до закладеного алгоритму(сценарієм)/пов'язані із Центральним процесором RS-485 інтерфейсом.

– Банки ІЧ команд. Випромінювання ІЧ команд із пам'яті контролера, раннє туди записаних. Керування апаратурою що має ІЧ інтерфейс.

Управляються центральним процесором або по командах від керуючих сенсорних панелей або відповідно до закладеного алгоритму/пов'язані із центральним процесором RS-485 інтерфейсом.

Отже, для того щоб зробити розумний будинок потрібно:

– Зробити схему розташування периферійного встаткування на об'єкті, робиться на поетажних планах.

– Пронумерувати всі приміщення в будинку.

– Скласти список проводів/тип кабелю, маркування, звідки й куди йде проведення, які повинні підходити до кожному елементу системи кабельний журнал.

– Відкрити вікно “Редактор системи управління домом” і мишкою створити необхідні елементи системи спочатку сторінки кімнат, потім на цих сторінках і інші елементи керування будинком. Під час роботи в редакторі ви створюєте графічний інтерфейс керування системи.

– Тепер залишилося додати графічному інтерфейсу індивідуальний дизайн і підключити до центрального процесора необхідне встаткування/список цього встаткування також формується в процесі роботи редактора.

3.3 Розробка функціональної схеми

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

– 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, тобто дозволяють через розроблене програмне забезпечення за допомогою пультів, або персонального комп’ютера управляти усіма блоками 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.ін.).

Розглянемо роботу цих блоків більш детально.

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.

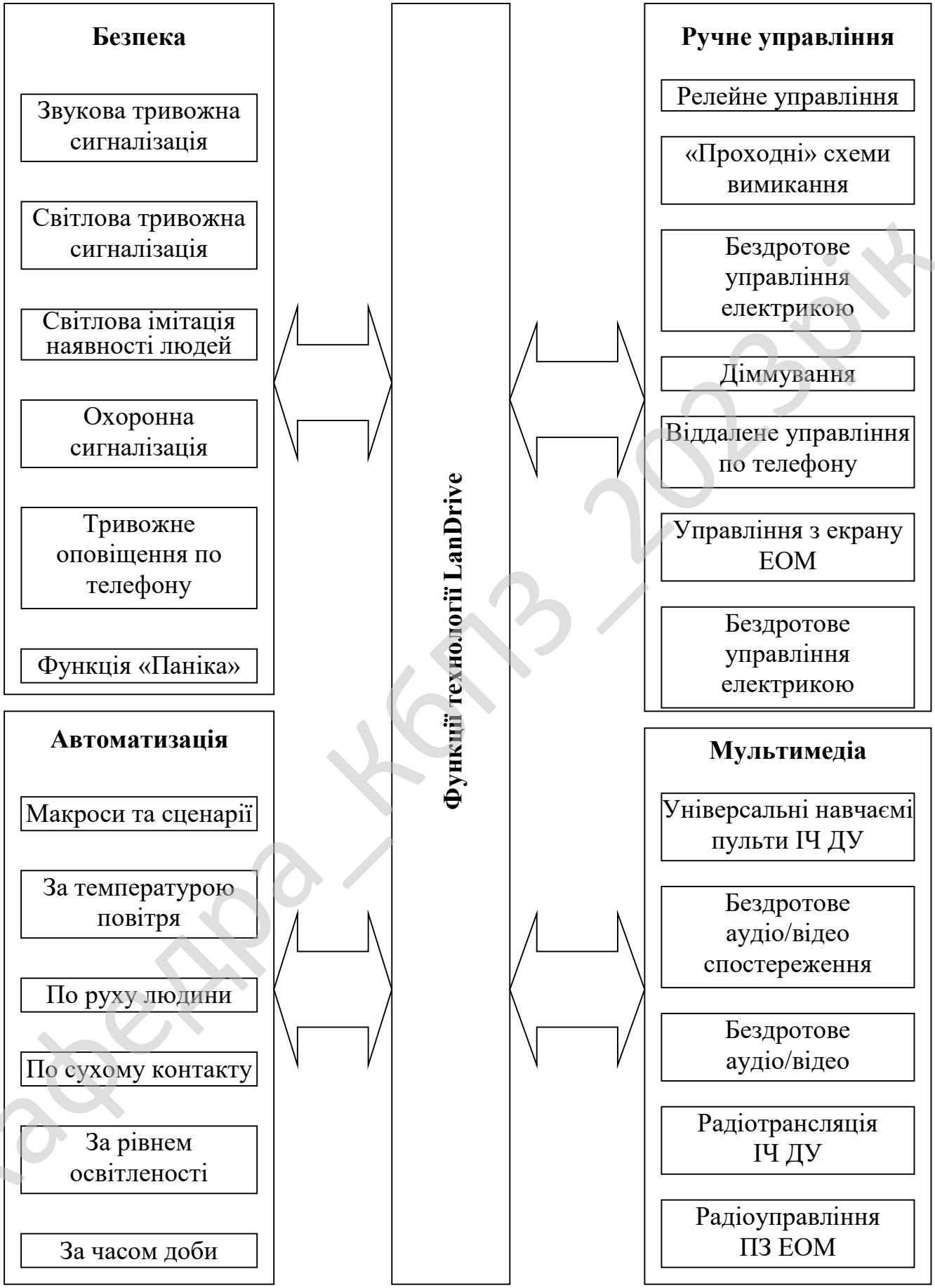

Рисунок 3.2 – Функціональна схема системи

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.

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.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. Першим процесом, який запускається у системі, є процес виведення головного вікна програми.

Цей процес взаємодіє з процесом ввімкнення LanDrive Configurator.

Процес ввімкнення LanDrive Configurator взаємодіє з наступними процесами:

– Процес читання показників пристроїв.

– Процес керування пристроями будинку.

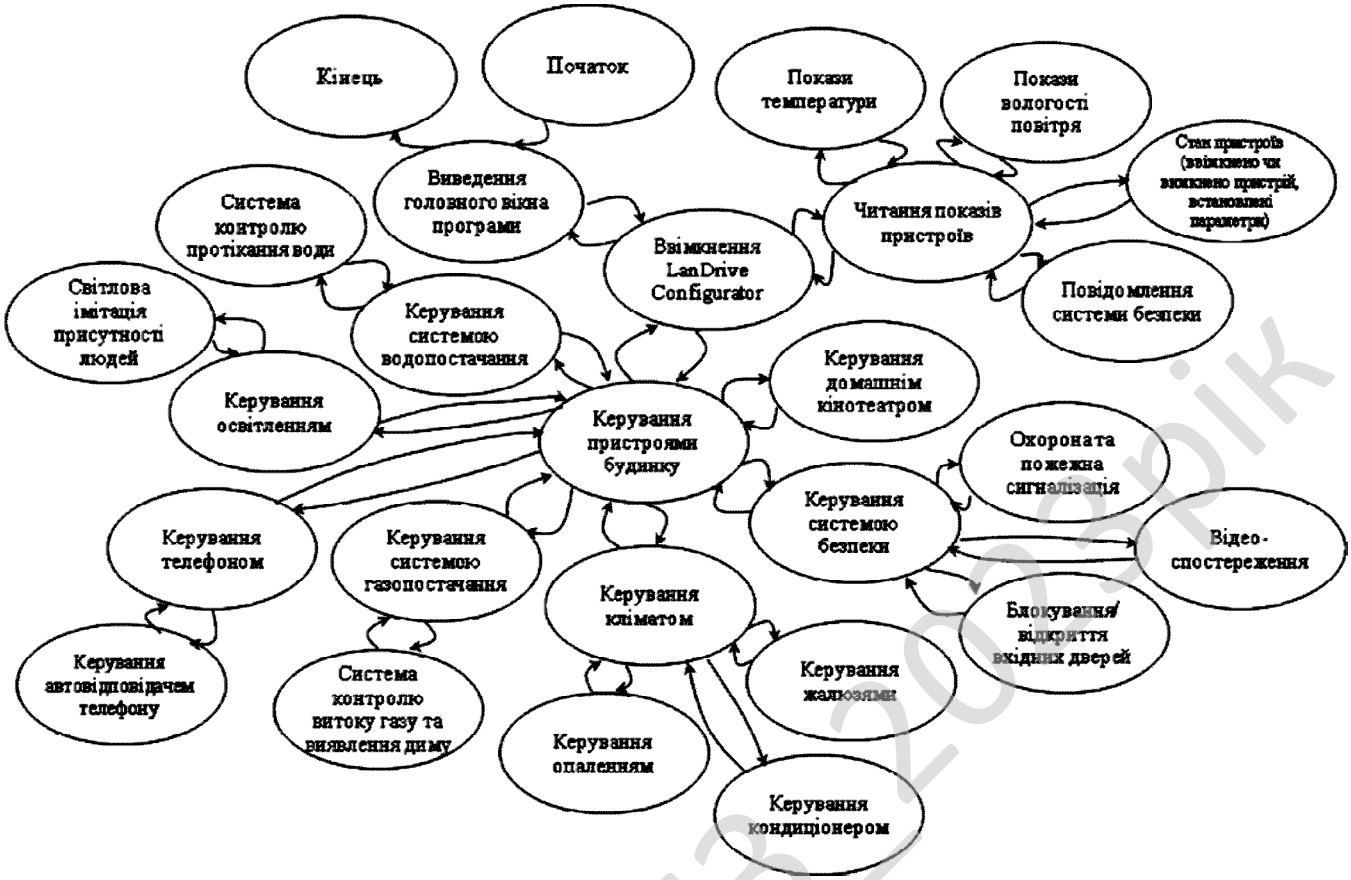

Рисунок 3.3 – Діаграма взаємодії процесів

Процес читання показників пристроїв взаємодіє з наступними процесами:

– Процес показу температури.

– Процес показу вологості повітря.

– Процес стану пристроїв.

– Процес повідомлення системи безпеки.

Процес керування пристроями будинку взаємодіє з наступними процесами:

– Процес керування системою водопостачання, який взаємодіє з процесом системи контролю протікання води.

– Процес керування системою освітлення, який взаємодіє з процесом системи світлової імітації присутності людей.

– Процес керування телефоном, який взаємодіє з процесом керування автовідповідачем телефону.

– Процес керування системою газопостачання, який взаємодіє з процесом системи контролю витоку газу та виявлення диму.

– Процес керування кліматом.

– Процес керування системою безпеки.

– Процес керування домашнім кінотеатром.

Процес керування кліматом взаємодіє з наступними процесами:

– Процес керування опаленням.

– Процес керування кондиціонером.

– Процес керування жалюзі.

Процес керування системою безпеки взаємодіє з наступними процесами:

– Процес охоронної та пожежної сигналізації.

– Процес відеоспостереження.

– Процес блокування/відкриття вхідних дверей.

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 РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього користувач обирає яку дію йому робити:

– Управління освітленням.

– Управління домашнім кінотеатром.

– Управління водопостачанням.

– Управління газопостачанням.

– Управління кліматом.

– Управління телефонним зв’язком.

Якщо користувач обирає керування освітленням, тоді виконуються наступні дії:

– Визначення поточного стану освітлення.

– Виведення поточного стану освітлення.

– Запускається підпрограма керування освітленням та його яскравістю.

Якщо користувач обирає керування домашнім кінотеатром, тоді запускається підпрограма керування домашнім кінотеатром.

Якщо користувач обирає керування водопостачанням, тоді виконуються наступні дії:

– Визначення поточного стану водопостачання.

– Виведення поточного стану водопостачання.

– Запускається підпрограма керування системою контролю підтікання води, клапанами холодної та гарячої води.

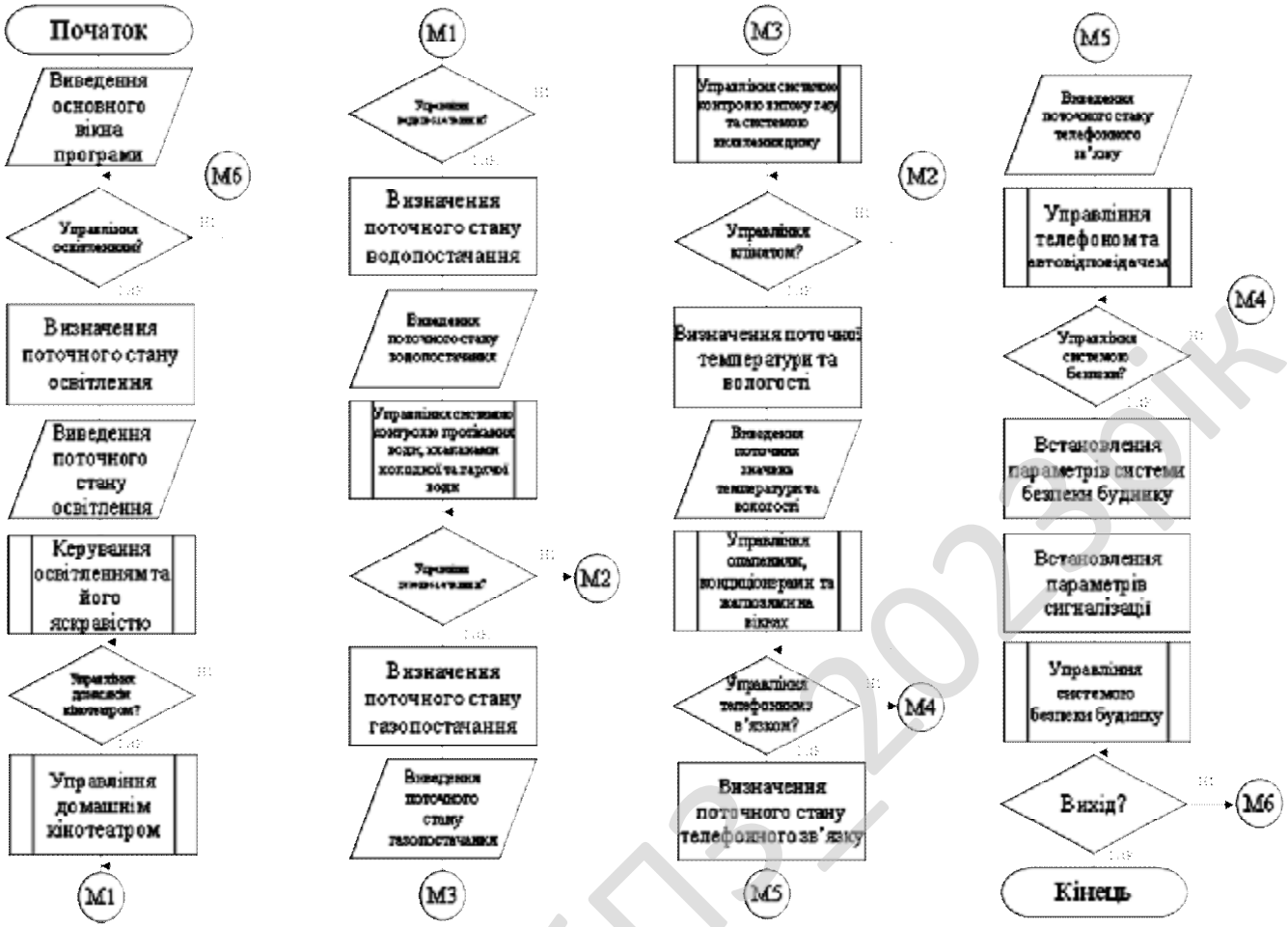

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає керування газопостачанням, тоді виконуються наступні дії:

– Визначення поточного стану газопостачання.

– Виведення поточного стану газопостачання.

– Запускається підпрограма керування системою контролю витоку газу та системою виявлення диму.

Якщо користувач обирає керування кліматом, тоді виконуються наступні дії:

– Визначення поточного стану температури та вологості.

– Виведення поточного стану температури та вологості.

– Запускається підпрограма керування опаленням, кондиціюванням та жалюзями на вікнах.

Якщо користувач обирає керування телефонним зв’язком, тоді виконуються наступні дії:

– Визначення поточного стану телефонного зв’язку.

– Виведення поточного стану телефонного зв’язку.

– Запускається підпрограма керування телефоном та автовідповідачем.

Якщо користувач обирає керування системою безпеки, тоді виконуються наступні дії:

– Встановлення параметрів системи безпеки будинку.

– Встановлення параметрів системи сигналізації.

– Запускається підпрограма керування системою безпеки будинку.

Після виконання усіх вищеперерахованих дій, користувач обирає працювати йому далі з системою, або ні.

На рисунку 4.2 зображена блок-схема роботи підпрограми кібербезпеки для захищеного управління будинком на основі технології LanDrive.

Підпрограма управління працює наступним чином:

– Вказується адреса пристрою.

– Вказується номер функції.

– Вказуються дані для передачі.

– Обчислюється контрольна сума кадру.

– Передається кадр, який містить у собі наступні дані: адресу пристрою, номер функції, дані, контрольну суму.

Якщо отримана команда є командою керування пристроєм, тоді виконуються наступні дії:

– Виконується команда.

– Виводяться дані про результат виконання команди.

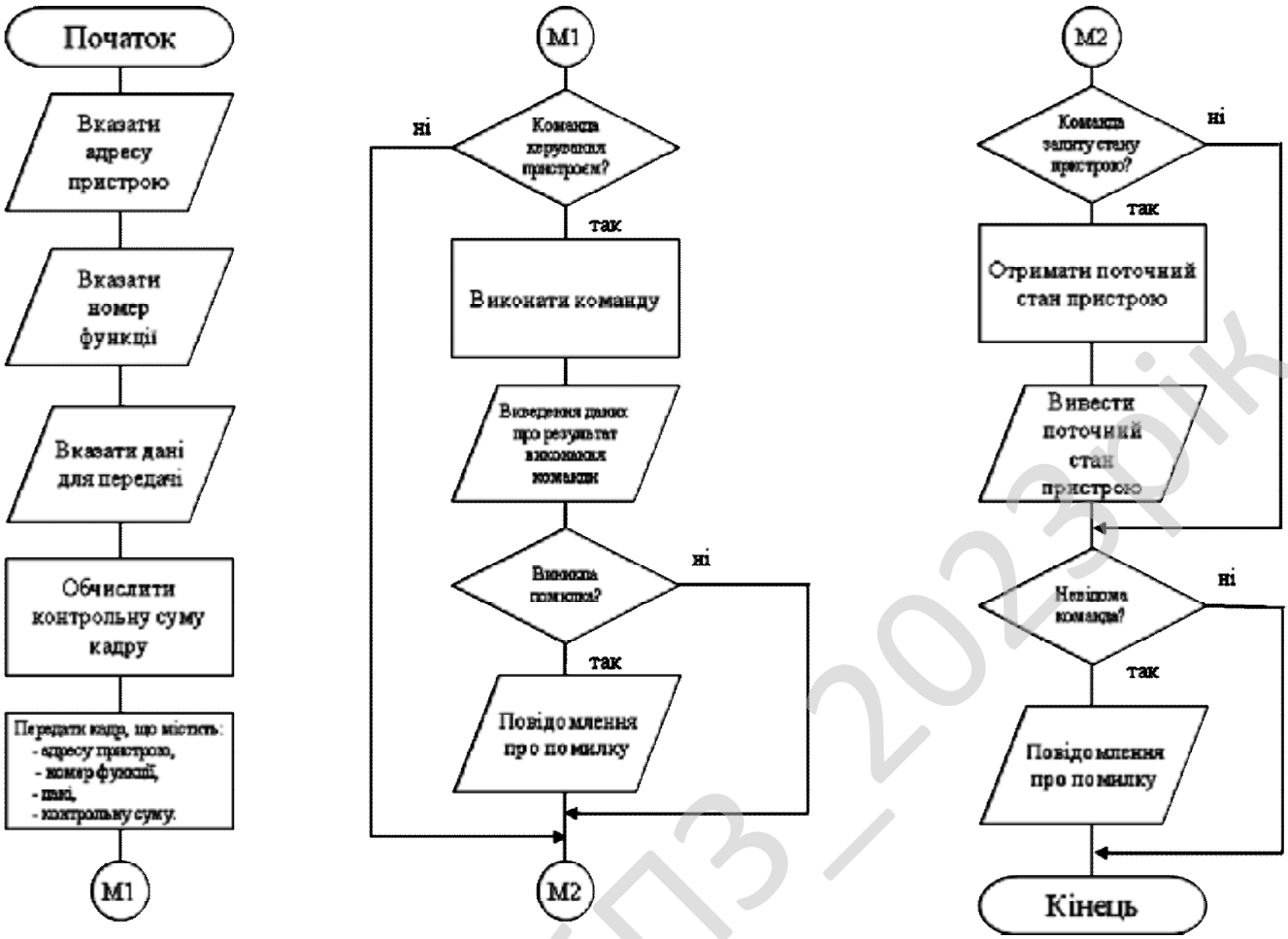

Рисунок 4.2 – Блок-схема роботи підпрограми кібербезпеки для захищеного управління будинком на основі технології LanDrive

Якщо виникла помилка, тоді виводиться повідомлення про помилку.

Якщо отримана команда є командою запису стану пристрою, тоді виконуються наступні дії:

– Отримується поточний стан пристрою.

– Виводиться поточний стан пристрою.

Якщо, команда, яка отримана, є невідомою, тоді виводиться повідомлення про помилку.

Для взаємодії на прикладному рівні використовується широко відомий протокол Modbus/RTU.

Modbus – комунікаційний протокол, заснований на архітектурі «клієнт-сервер». Широко застосовується в промисловості для організації зв'язку між електронними пристроями. Може використовувати для передачі даних через послідовні лінії зв'язку RS-485, RS-422, RS-232, а також мережі TCP/IP (Modbus TCP). Modbus був розроблений компанією Modicon (в даний час належить Schneider Electric) для використання в її контролерах з програмованою логікою. Вперше специфікація протоколу була опублікована в 1979 році. [1] Це був відкритий стандарт, що описує формат повідомлень і способи їх передачі в мережі складається з різних електронних пристроїв. Спочатку контролери MODICON використовували послідовний інтерфейс RS-232. Пізніше став застосовуватися інтерфейс RS-485, так як він забезпечує більш високу надійність, дозволяє використовувати довші лінії зв'язку і підключати до однієї лінії кілька пристроїв. Багато виробників електронного устаткування підтримали стандарт, на ринку з'явилися сотні використовують його виробів. В даний час розвитком Modbus займається некомерційна організація Modbus-IDA, створена виробниками та користувачами електронних приладів.

Modbus відноситься до протоколів прикладного рівня мережевої моделі OSI. Контролери на шині Modbus взаємодіють, використовуючи клієнт-серверну модель, засновану на транзакція х, що складаються із запиту і відповіді.

Зазвичай в мережі є тільки один клієнт, так зване, «головний» (англ. *master*) пристрій, і кілька серверів – «підлеглих» (англ. *slaves*) пристроїв. Головне пристрій ініціює транзакції (передає запити). Підлеглі пристрої передають запитувані головним пристроєм дані, або виробляють запитувані дії. Головний може адресуватися індивідуально до підлеглого або ініціювати передачу широкомовного повідомлення для всіх підлеглих пристроїв. Підлеглий пристрій формує повідомлення і повертає його у відповідь на запит, адресований саме йому. При отриманні широкомовного запиту відповідь не формується.

Специфікація Modbus описує структуру запитів і відповідей. Їх основа – елементарний пакет протоколу, так званий PDU (Protocol Data Unit). Структура

PDU не залежить від типу лінії зв'язку і включає в себе код функції і поле даних. Код функції кодується однобайтові полем і може приймати значення в діапазоні 1 ... 127. Діапазон значень 128 ... 255 зарезервований для кодів помилок. Поле даних може бути змінної довжини. Розмір пакета PDU обмежений 253 байтами.

Modbus PDU	
номер функції	дані
1 байт	N <253 (байт)

Рисунок 4.3 – Пакет Modbus PDU

Для передачі пакету по фізичних лініях зв'язку PDU поміщається в інший пакет, що містить додаткові поля. Цей пакет має назву ADU (Application Data Unit). Формат ADU залежить від типу лінії зв'язку.

Існують три основні реалізації протоколу Modbus, дві для передачі даних по послідовних лініях зв'язку, як мідним EIA/TIA-232-E (RS-232), EIA-422, EIA/TIA-485-A (RS -485), так і оптичним і радіо:

– Modbus ASCII – для обміну використовуються тільки ASCII символи. Для перевірки цілісності використовується алгоритм en: Longitudinal redundancy check. Повідомлення поділяється на стовпці за допомогою символу «:» і закінчується символами нового рядка CR / LF.

– Modbus RTU.

І для передачі даних по мережах Ethernet поверх TCP/IP:

– Modbus TCP.

Загальна структура ADU наступна (у залежності від реалізації, деякі з полів можуть бути відсутні).

адреса відомого пристрою	код функції	дані	блок виявлення помилок

Рисунок 4.4 – Загальна структура ADU

Де:

– Адреса відомого пристрою – адреса підлеглого пристрою, до якого адресовано прохання. Відомі пристрої відповідають тільки на запити, що надійшли на їх адресу. Відповідь також починається з адреси відповідає відомого пристрою, який може змінюватися від 1 до 247. Адреса 0 використовується для широкомовної передачі, його розпізнає кожне пристрій, адреси в діапазоні 248 ... 255 – зарезервовані.

– Номер функції – це наступне однобайтне поле кадру. Воно говорить відомому пристрою, які дані або виконання якого дії вимагає від нього ведучий пристрій.

– Дані – поле містить інформацію, необхідну відомому пристрою для виконання заданої майстром функції або містить дані, що передаються веденим пристроєм у відповідь на запит ведучого. Довжина і формат поля залежить від номера функції.

– Блок виявлення помилок – контрольна сума для перевірки відсутності помилок в кадрі.

Максимальний розмір ADU для послідовних мереж RS232/RS485 – 256 байт, для мереж TCP – 260 байт.

Для Modbus TCP ADU виглядає наступним чином.

ід транзакції	ід протоколу	довжина пакету	адреса відомого пристрою	код функції	дані

Рисунок 4.5 – Загальна структура ADU для Modbus TCP

Де:

– Ід транзакції – два байти, зазвичай нулі.

– Ід протоколу – два байти, нулі.

– Довжина пакету – два байти, старший потім молодший, довжина наступної за цим полем частини пакета.

– Адреса відомого пристрою – адреса підлеглого пристрою, до якого адресовано прохання. Зазвичай ігнорується, якщо з'єднання встановлено з певним пристроєм. Може використовуватися, якщо з'єднання встановлено з мостом, який виводить нас, наприклад, в мережу RS485.

Поле контрольної суми в Modbus TCP відсутнє.

Категорії кодів функцій

У діючій в даний час специфікації протоколу визначаються три категорії кодів функцій:

– Стандартні команди. Їх опис має бути опублікована та затверджено Modbus-IDA. Ця категорія включає в себе як вже певні, так і вільні в даний час коди.

– Користувальницькі команди. Два діапазони кодів (від 65 до 72 і від 100 до 110), для яких користувач може реалізувати довільну функцію. При цьому не гарантується, що якийсь інший пристрій не буде використовувати той же самий код для виконання іншої функції.

– Зарезервовані. У цю категорію входять коди функцій, які не є стандартними, але вже використовуються в пристроях, що виробляються різними компаніями. Це коди 9, 10, 13, 14, 41, 42, 90, 91, 125, 126 і 127.

Модель даних

Одне з типових застосувань протоколу – читання і запис даних в регістри контролерів. Специфікація протоколу визначає чотири таблиці даних.

Таблиця 4.1 – Таблиця даних

Таблиця	Тип елемента	Тип доступу
Дискретні входи (англ. *Discrete Inputs*)	Один біт	тільки читання
Регістри прапорів (англ. *Coils*)	Один біт	читання і запис
Регістри введення (англ. *Input Registers*)	16-бітне слово	тільки читання
Регістри зберігання (англ. *Holding Registers*)	16-бітне слово	читання і запис

Доступ до елементів в кожній таблиці здійснюється за допомогою 16-бітного адреси, першій клітинці відповідає адреса 0. Таким чином, кожна таблиця може містити до 65536 елементів. Специфікація не визначає, що фізично повинні представляти собою елементи таблиць і по яким внутрішнім адресами пристрою вони повинні бути доступні. Наприклад, припустимо організувати перекриваються таблиці, У цьому випадку команди працюють з дискретними даними і з 16-бітними регістрами будуть фактично звертатися до одних і тих же даних.

PDU запиту і відповіді для стандартних функцій							
Номер функції	запит / відповідь						
1 (0x01)	A_1	A_0	Q_1	Q_0			
	N	D (N байт)					
2 (0x02)	A_1	A_0	Q_1	Q_0			
	N	D (N байт)					
3 (0x03)	A_1	A_0	Q_1	Q_0			
	N	D (N байт)					
4 (0x04)	A_1	A_0	Q_1	Q_0			
	N	D (N байт)					
5 (0x05)	A_1	A_0	D_1	D_0			
	A_1	A_0	D_1	D_0			
6 (0x06)	A_1	A_0	D_1	D_0			
	A_1	A_0	D_1	D_0			
15 (0x0F)	A_1	A_0	Q_1	Q_0	N	D (N байт)	
	A_1	A_0	Q_1	Q_0			
16 (0x10)	A_1	A_0	Q_1	Q_0	N	D (N байт)	
	A_1	A_0	Q_1	Q_0			

Рисунок 4.6 – Стандартні функції протоколу Modbus

– A_1 і A_0 – адреса елемента.

– Q_1 і Q_0 – кількість елементів.

– N – кількість байт даних.

– D – дані.

Слід зазначити, що зі способом адресації даних пов'язана певна плутанина. Modbus був спочатку розроблений для контролерів Modicon. У цих контролерах для кожної з таблиць використовувалася спеціальна нумерація. Наприклад, першому регістру введення відповідав номер комірки 30001, а першому регістру зберігання – 40001. Таким чином, регістру зберігання з адресою 107 в команді Modbus відповідав регістр № 40108 контролера. Хоча така відповідність адрес більше не є частиною стандарту, деякі програмні пакети можуть автоматично «коригувати» вводяться користувачем адреси, наприклад, віднімаючи 40001 з адреси регістра зберігання.

Читання даних

Для читання значень з перерахованих вище таблиць даних використовуються функції з кодами 1-4 (шістнадцяткові значення 0x01-0x04):

– 1 (0x01) – читання значень з декількох регістрів прапорів (Read Coil Status).

– 2 (0x02) – читання значень з декількох дискретних входів (Read Discrete Inputs).

– 3 (0x03) – читання значень з декількох регістрів зберігання (Read Holding Registers).

– 4 (0x04) – читання значень з декількох регістрів введення (Read Input Registers).

Запит складається із адреси першого елемента таблиці, яку потрібно прочитати, і кількості зчитувальних елементів. Адреса та кількість даних задаються 16-бітними числами, старший байт кожного з них передається першим.

У відповіді передаються запитані дані. Кількість байт даних залежить від кількості замовлених елементів. Перед даними передається один байт, значення якого дорівнює кількості байт даних.

Значення регістрів зберігання і регістрів введення передаються починаючи із зазначеної адреси, по два байти на регістр, старший байт кожного регістру передається першим.

байт 1	байт 2	байт 3	байт 4	...	байт N-1	байт N
$R_{A,1}$	$R_{A,0}$	$R_{A+1,1}$	$R_{A+1,0}$	...	$R_{A+Q-1,1}$	$R_{A+Q-1,0}$

Рисунок 4.7 – Значення регістрів зберігання і регістрів введення

Значення прапорів і дискретних входів передаються в упакованому вигляді: по одному біту на прапор. Одиниця означає включений стан, нуль – вимкнений. Значення запитаних прапорів заповнюють спочатку перший байт, починаючи з молодшого біта, потім наступні байти, також від молодшого біта до старших. Молодший біт першого байта даних містить значення прапора, зазначеного в полі «адреса». Якщо запитано кількість прапорів, не кратну восьми, то значення додаткового біта заповнюються нулями.

байт 1								...	байт N						
F_{A+7}	F_{A+6}	F_{A+5}	F_{A+4}	F_{A+3}	F_{A+2}	F_{A+1}	F_{A}	...	0	...	0	F_{A+Q-1}	F_{A+Q-2}	...	

Рисунок 4.8 – Значення прапорів і дискретних входів

Запис одного значення:

– 5 (0x05) – запис значення одного прапора (Force Single Coil).

– 6 (0x06) – запис значення в один регістр зберігання (Preset Single Register).

Команда складається з адреси елемента (2 байти) і встановлюваного значення (2 байти). Для регістру зберігання значення є просто 16-бітним словом.

Для прапорів значення 0xFF00 означає включений стан, 0x0000 – вимкнений, інші значення неприпустимі.

Якщо команда виконана успішно, ведене пристрій повертає копію запиту.

Запис декількох значень:

– 15 (0x0F) – запис значень в кілька регістрів прапорів (Force Multiple Coils).

– 16 (0x10) – запис значень в кілька регістрів зберігання (Preset Multiple Registers).

Команда складається з адреси елемента, кількості змінюваних елементів, кількості переданих байт встановлюваних значень і самих встановлюваних значень. Дані упаковуються так само, як в командах читання даних.

Відповідь складається з початкової адреси і кількості змінених елементів.

Нижче наведено приклад команди ведучого пристрою і відповіді веденого (для Modbus RTU).

Напрямок передачі	00 Адреса підлеглого пристрою	01 Номер функції	02 Адреса ст. байт	03 Адреса мол. байт	04 Кількість прапорів ст. байт	05 Кількість прапорів мол. байт	06 Кількість байт даних	07 Дані (значення для прапорів біти 0-7)	08 Дані (значення для прапорів біти 8-15)	09 CRC мол. байт	0A CRC ст. байт
Master → Slave	0x01	0x0F	0x00	0x13	0x00	0x0A	0x02	0xCD	0x01	0x72	0xCB

Рисунок 4.9 – Приклад команди ведучого пристрою

Напрямок передачі	00 адреса підлеглого пристрою	01 номер функції	02 Адреса ст. байт	03 Адреса мл. байт	04 Кількість прапорів ст. байт	05 Кількість прапорів мл. байт	05 CRC мл. байт	06 CRC ст. байт
Slave → Master	0x01	0x0F	0x00	0x13	0x00	0x0A	0x24	0x09

Рисунок 4.10 – Приклад відповіді веденого пристрою

Контроль помилок у протоколі Modbus RTU

Під час обміну даними можуть виникати помилки двох типів:

– Помилки, пов'язані з спотвореннями при передачі даних.

– Логічні помилки.

Помилки першого типу виявляються за допомогою фреймів символів, контролю парності і циклічної контрольної суми CRC-16-IBM (використовується число-поліном = 0xA001). При цьому молодший байт передається першим, на відміну від байтів адреси і значення регістра в PDU

RTU фрейм

У RTU режимі повідомлення має починатися і закінчуватися інтервалом тиші – часом передачі не менше 3.5 символів при даній швидкості в мережі. Першим полем потім передається адреса пристрою.

Слідом за останнім переданим символом також слід інтервал тиші тривалістю не менше 3.5 символів. Нове повідомлення може починатися після цього інтервалу.

Фрейм повідомлення передається безперервно. Якщо інтервал тиші тривалістю 1.5 виник під час передачі фрейму, приймаючий пристрій повинен ігнорувати цей фрейм як неповний.

Таким чином, нове повідомлення повинно починатися не раніше 3.5 інтервалу, так як в цьому випадку встановлюється помилка.

Трохи про інтервали (мова йде про Serial Modbus RTU): при швидкості 9600 і 11 бітах в кадрі (стартовий біт + 8 біт даних + біт контролю парності + стоп-біт): 3.5 * 11 / 9600 = 0,00401041 (6), тобто більше 4 мс; 1.5 * 11 / 9600 = 0,00171875, тобто більше 1 мс. Для швидкостей більше 19200 бод допускається використовувати інтервали 1,75 і 0,75 мс відповідно.

Логічні помилки

Для повідомлень про помилки другого типу протокол Modbus RTU передбачає, що пристрої можуть відсилати відповіді, що свідчать про помилкову ситуації. Ознакою того, що відповідь містить повідомлення про помилку, є встановлений старший біт коду команди. Приклад кадру при виявленні помилки веденим пристроєм, у відповідь на запит наведено на рисунку 4.11.

1. Якщо Slave приймає коректний запит і може його нормально обробити, то повертає стандартний відповідь.

2. Якщо Slave не приймає будь-якого значення, ніякої відповіді не відправляється. Master діагностує помилку по тайм-ауту.

3. Якщо Slave приймає запит, але виявляє помилку (parity, LRC, or CRC), ніякої відповіді не відправляється. Master діагностує помилку по тайм-ауту.

4. Якщо Slave приймає запит, але не може його обробити (звернення до неіснуючого регістру і т. д.), відправляється відповідь містить у собі дані про помилку.

Напрямок передачі	Адреса підлеглого пристрою	Номер функції	Дані (або код помилки)	CRC
Запит (Master → Slave)	0x01	0x77	0xDD	0xC7 0xA9
Відповідь (Slave → Master)	0x01	0xF7	0xEE	0xE6 0x7C

Рисунок 4.11 – Кадр відповіді (Slave → Master) при виникненні помилки modbus RTU

Стандартні коди помилок:

– 01 – Прийнятий код функції не може бути опрацьований на підпорядкованому.

– 02 – Адреса даних, вказаний у запиті, не доступний даному підлеглому.

– 03 – Величина, що міститься в полі даних запиту, є неприпустимою величиною для підлеглого.

– 04 – Невідновлювальна помилка мала місце, поки підлеглий намагався виконати затребувану дію.

– 05 – Підлеглий прийняв запит і обробляє його, але це вимагає багато часу. Ця відповідь оберігає головного від генерації помилки тайм-ауту.

– 06 – Підлеглий зайнятий обробкою команди. Головний повинен повторити повідомлення пізніше, коли підлеглий звільниться.

– 07 – Підлеглий не може виконати програмну функцію, прийняту в запиті. Цей код повертається для невдалого програмного запиту, що використовує функції з номерами 13 або 14. Головний повинен запросити діагностичну інформацію або інформацію про помилки з підлеглого.

– 08 – Підлеглий намагається читати розширену пам'ять, але виявив помилку паритету. Головний може повторити запит, але звичайно в таких випадках потрібний ремонт.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i <<< 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і ⊕ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} <<< 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Блок меню.

– Блок вибору системою, якою потрібно керувати.

– Блок керування елементами обраної системи.

Блок меню складається з наступних елементів:

– Освітлення.

– Водопостачання.

– Газопостачання.

– Клімат-контроль.

– Телефон.

– Мультимедіа.

– Безпека.

– Параметри.

– Довідка.

У інших блоках деталізуються та візуалізуються елементи управління тим, або іншим приладом, або системою.

На рисунку 5.2 зображено вікно довідки, з якого є можливість отримати наступні дані:

– Розробник проекту.

– Керівник проекту.

– Тема проекту.

– Місце розробки проекту.

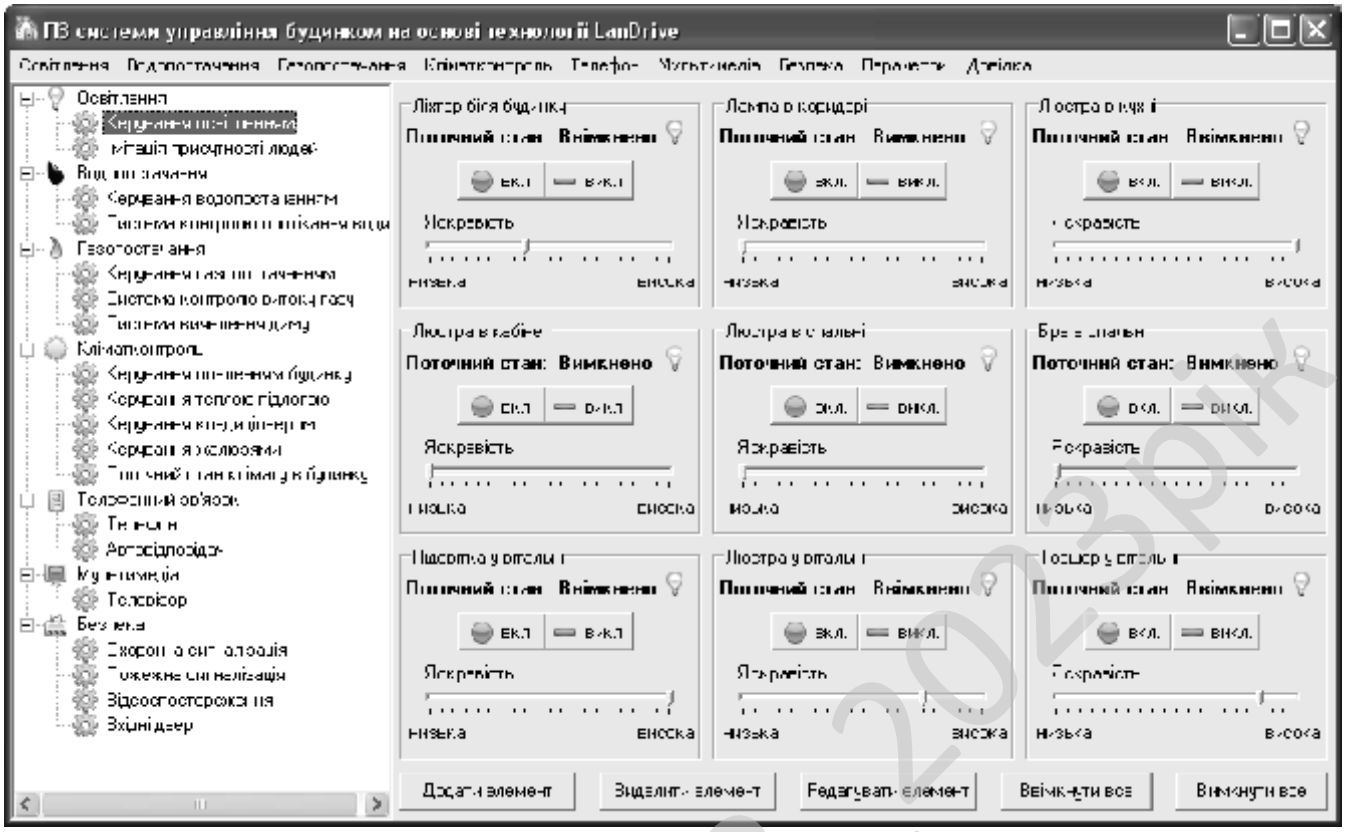

Рисунок 5.1 – Головне вікно програми

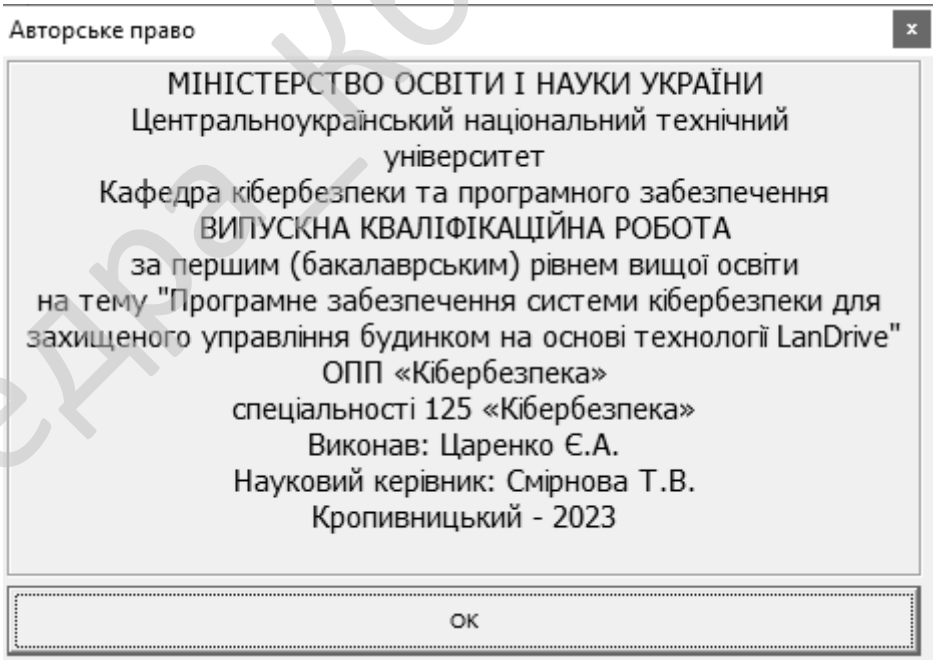

Рисунок 5.2 – Довідка про програму

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для захищеного управління будинком на основі технології LanDrive.

– Досліджена система для захищеного управління будинком на основі технології LanDrive.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для захищеного управління будинком на основі технології LanDrive.

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 призначені для системи кібербезпеки для захищеного управління будинком на основі технології

LanDrive. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

2. Свами М.Н., Тхуласираман К. Графы, сети и алгоритмы: пер. с англ. / М.Н. Свами, К. Тхуласираман; под ред. В.А. Горбатова. – М.: Мир, 1984. – 454 с.

3. Семенов А.Б. Структурированные кабельные системы / А.Б. Семенов, С.К. Стрижаков, И.Р. Сунчелей – 3-е изд. – М.: “Компьютер-Пресс”, 2001. – 608с.

4. Семенов А. Д. Идентификация объектов управления: Учебн. Пособие / А.Д.Семенов, Д.В.Артамонов, А.В.Брюхачев - Пенза: Изд-во Пенз. гос. ун-та, 2003.- 211 с.

5. Семенов С.Г. Анализ методов прогнозирования в телекоммуникационных сетях автоматизированных систем управления / С.Г.Семенов // Збірник наукових праць «Системи управління, навігації та зв’язку», - К.:ЦНДІ навігації і управління, - 2008.-Вип. 2(6) .- С.134-137

6. Семенов С.Г. Математическая модель процесса доставки информационных пакетов в компьютерной сети системы критического применения / С.Г.Семенов, И.В.Ильина // Науково-технічний журнал «Радіоелектронні і комп’ютерні системи» Х.:ХАІ, - 2008.-Вип. 1(28) - С.162-165

7. Семенов С.Г. Оптимизация трафика на основе сбалансированной загрузки информационно-телекоммуникационной сети // Системи обробки інформації. – Х.: ХВУ, 2004. – № 8(36). – С.206-210

8. Семенов С.Г. Сравнительные исследования методов идентификации трафика в телекоммуникационной сети для повышения оперативности передачи данных / С.Г.Семенов, Е.В.Мелешко // Науково-технічний журнал "Прикладная радиоэлектроника" Том 9 №3. - Х: ХНУРЕ. – 2010. – С.444-448.

9. Семенов С.Г. Сравнительные исследования и анализ алгоритмов

управления очередями в многопротокольных узлах связи телекоммуникационной сети / С.Г.Семенов, Е.В.Мелешко, В.В.Босько // «Новітні технології – для захисту повітряного простору» Матеріали шостої наукової конференції 14-15 квітня. – Х: ХУПС, 2010.– С.132.

10. Semenov S. The method of processing and identification of telecommunication traffic based on BDS-tests / S. Semenov, A.Smirnov., E.Meleshko // The book of materials International Conference «Statistical Methods of Signal and Data Processing (SMSDP-2010)» –Kiev, Ukraine, National Aviation University "NAU-Druk" Publishing House, October 13-14, 2010. – С.166-168. – engl.

11. Семенов Ю.А. Сети Інтернет. Архитектура и протоколы / Ю.А. Семенов. – М.: Блик плюс, 1998. – 424 с.

12. Смирнов А.А. Разработка методики оценки среднего времени обслуживания информационных пакетов в телекоммуникационной сети / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи управління, навігації та зв'язку. – Київ: ДП «Центральний науково-дослідний інститут навігації і управління», 2009. – Вип. 2(10). – С.162-165.

13. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

14. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011. – 193-195 с.

15. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

16. Столлингс В. Современные компьютерные сети / Вильям

Столлингс.– СПб.: Питер, 2003. – 778 с.

17. Сэломон Д. Сжатие данных, изображений и звука / Д. Сэломон. – М.: Техносфера, 2004. – 368 с.

18. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. — 848 с.

19. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

20. Уолрэнд Дж. Телекоммуникационные и компьютерные сети / Дж. Уолрэнд. – М.: Постмаркет, 2001. – 480 с.

21. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

22. Хаусли Т. Системы передачи и телеобработки данных: пер. с англ. / Т. Хаусли; под ред. Ю.М. Мартынова. – М.: Радио и связь, 1994. – 452 с.

23. Чернявский Г.М. Новые технологии в спутниковых системах / Г.М. Чернявский // Информационные технологии и вычислительные системы. – М.: Институт микропроцессорных вычислительных систем РАН, 2005. – Вып.1 – С. 3 – 11.

24. Шелевицький І.В. Методи та засоби сплайн-технології обробки сигналів складної форм / І.В. Шелевицький– Кривий Ріг: Європейський університет, 2002. - 304 с.

25. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

26. Desrochers, S. Mohseni On determining the structure of non-linear systems // Int. J. Control. 1984. V. 40. N 5. — P. 923-938.

27. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

28. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.

29. An Chen, Albert Kai-Sun Wong, and Chin-Tau Lea, Senior Member. Routing and Time-Slot Assignment in Optical TDM Networks // IEEE Journal on selected areas in communications. – 2004. – №22(9), P. 1648-1657.

30. Andrew B. Kahng, Stefanus Mantik, and Dirk Stroobandt. Toward Accurate Models of Achievable Routing // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2001. – №20(5), P. 648-659.

31. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing // IEEE/ACM Transactions on Networking. – 2001. – №9(2), P. 162-176.

32. Basabi Chakraborty. Simultaneous Search for Multiple Routes using Genetic Algorithm / IEEE lntemational Conference on Computational oltelligencs for Measurement System and Applications Boston. MA, USA, 14-16, July 2004, P. 77-80/

33. LeBaron "A Fast Algorithm for the BDS Statistic", Studies in Nonlinear Dynamics and Econometrics. 1997. Vol. 2. No. 2. P. 53-59.

34. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: a survey // IEEE Communications Magazine. – 2000. – №38(1). – P. 40 - 46.

35. Casetti, R. Lo Cigno, M. Mellia, M. Munafo. A New Class of QoS Routing Strategies Based on Network Graph Reduction // Proceedings of IEEE INFOCOM. – 2002, P.715-722.

36. Chiara Francalanci and Paolo Giacomazzi. High-Performance Self-Routing Algorithm for Multiprocessor Systems with Shuffle Interconnections // IEEE Transactions on parallel and distributed systems. – 2006. – №1, P. 38-50.

37. Chris Loeser, Andre Brinkmann, Ulrich Ruckert. Distributed Path Selection (DPS) a Traffic Engineering Protocol for IP-Networks / Proceedings of the 37th Hawaii International Conference on System Sciences – 2004, P. 1-8.

38. Dai Boong Lee and Hwangjun Song. Dynamic Class Selecting Mechanism for Guaranteed Service with Minimum Cost over Relative Differentiated-Services Networks / IEEE International Conference on Multimedia and Expo (ICME) – 2004, P. 237-240.

39. Dan Pei, Dan Massey, Lixia Zhang. Detection of Invalid Routing Announcements in RIP Protocol // IEEE GLOBECOM – 2003, P. 1450-1455.

40. Donna Ghosh, Venkatesh Sarangan, and Raj Acharya. Quality-of-Service Routing in IP Networks // IEEE Transactions on Multimedia. – 2001. – 3(2), P. 200-208.

41. Chappell J. Padmore and C. Ellis. "A note on the distribution of BDS statistics for a real exchange rate series", Oxford Bulletin of Economics and Statistics, 58, 3, 561- 566, 1996.

42. Gang Cheng and Nirwan Ansari. A New Heuristics For Finding The Delay Constrained Least Cost Path // IEEE GLOBECOM – 2003, P. 3711-3715.

43. Gang Cheng, Li Zhu, and Nirwan Ansari. A New Deterministic Traffic Model for Core-Stateless Scheduling // IEEE Transactions on communications. – 2006. – № 4, P. 704-713.

44. G.Kreisselmeier A robust indirect adaptive control approach // Int. J. Control. 1986. Vol. 43. № 1. — P. 161–175.

45. G.Kreisselmeier, B.Anderson Robust // IEEE Trans. Autom. Control. 1987. Vol. AC-31. № 2. — P. 127-133.

46. G.Kreisselmeier, Narendra K. Stable model reference adaptive control in the presence of bounder disturbances // IEEE Trans. Autom. Control. 1982. Vol. AC-27. № 6. — P. 1169–1175.

47. Hui Ma, Dongfeng Wang, Farokh Bastani, I-Ling Yen, Kendra Cooper. A Model and Methodology for Composition QoS Analysis of Embedded Systems / Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'05) – 2005, P. 1-10.

48. H. Chaskar M. Eder, S. Nag "Considerations from the Service Management Research Group (SMRG) on Quality of Service (QoS) in the IP Network" //

RFC-3387. September 2002

49. J.P Crutchfield., B.S. McNamara Equations of motion from a data series // Complex Systems, 1987. V. 1. — P. 417-452.

50. J. K. J. Astrom, P. Eykhoff System Identification – A survey // Automatica. 1971. Vol. 7. № 2. — P. 123-162.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-125.23.0038.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Царенко Є.А.			Програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнова Т.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КБ-20-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 13-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для захищеного управління будинком на основі технології LanDrive;

– 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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 71 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки для захищеного управління будинком на основі технології LanDrive

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 47

Літера: РП

Кропивницький – 2023 року

Файл CPU_LanDrive_Socket.cpp – протокол LCN

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

#include <basepch.h>
#pragma hdrstop
#include "CPU_LanDrive_Socket.h"
#pragma package(smart_init)
//------- ------------------
//
static inline void ValidCtrCheck(TCPU_LanDrive_Socket *)
{
    new TCPU_LanDrive_Socket(NULL);
}
//#include "OcelotStateUnit.cpp"
//------- ------------------
__fastcall TCPU_LanDrive_Socket::TCPU_LanDrive_Socket(TComponent* Owner)
    : TClientSocket(Owner)
{
_LanDrive_StateChangeNum=new T_LanDrive_StateChangeNum;
OcelotStateChangeNum=new TOcelotStateChangeNum;
_LanDrive_State=new T_LanDrive_State;
OcelotState=new TOcelotState;

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

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

}

__fastcall TCPU_LanDrive_Socket::~TCPU_LanDrive_Socket(void)
{
delete _LanDrive_StateChangeNum;
delete OcelotStateChangeNum;
delete _LanDrive_State;
delete OcelotState;

}
//------- ------------------
namespace CPU_LanDrive_Socket
{
    void __fastcall PACKAGE Register()
    {
        TComponentClass classes[1] = {__classid(TCPU_LanDrive_Socket)};
        RegisterComponents(" CPU-XA", classes, 0);
    }
}
//------- ------------------
void __fastcall TCPU_LanDrive_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_LanDrive_Socket::MyOnDisconnect(System::TObject* Sender,
TCustomWinSocket* Socket)
```

```
{
SendBuf("d",1);
FAuth=2;
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,FAuth);
}

//------- -------------------
void __fastcall TCPU_LanDrive_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_LanDrive_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(&_LanDrive_StateChangeNum->Num,&local_pointer[1],2);
                _LanDrive_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_LanDrive_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_LanDrive_Socket::SendOcelotStatusQuery(void)
{
char buf[3];
buf[0]='q';
memcpy(&buf[1],&OcelotStateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
void __fastcall TCPU_LanDrive_Socket::Send10StatusQuery(void)
{
char buf[3];
buf[0]='x';
memcpy(&buf[1],&_LanDrive_StateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
unsigned short __fastcall TCPU_LanDrive_Socket::GetDataFromUnit(unsigned char
UnitNumber)
{
return OcelotState->GetDataFromUnit(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_LanDrive_Socket::GetIO(unsigned char
UnitNumber,unsigned char Point)
{
return OcelotState->GetIO(UnitNumber,Point);
}
//------- -------------------
unsigned char __fastcall TCPU_LanDrive_Socket::GetUnitVer(unsigned char
UnitNumber)
{
return OcelotState->GetUnitVer(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_LanDrive_Socket::GetUnitType(unsigned char
UnitNumber)
{
return OcelotState->GetUnitType(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPU_LanDrive_Socket::GetUnitsCount(void)
{
return OcelotState->GetUnitsCount();
}
//------- -------------------
TDateTime __fastcall TCPU_LanDrive_Socket::GetCPUXADateTime()
{
return OcelotState->GetCPUXADateTime();
}
//------- -------------------
unsigned short __fastcall TCPU_LanDrive_Socket::GetVariable(unsigned char
VarNumber)
{
return OcelotState->GetVariable(VarNumber);
}
//------- -------------------
unsigned short __fastcall TCPU_LanDrive_Socket::GetTimer(unsigned char
TimerNumber)
{
return OcelotState->GetTimer(TimerNumber);
}
//------- -------------------
```

```
TDateTime TCPU_LanDrive_Socket::GetTimerAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetTimer(VarNumber));
}
//------- -------------------
TDateTime TCPU_LanDrive_Socket::GetVarAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetVariable(VarNumber));
}
//------- -------------------
TDateTime TCPU_LanDrive_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_LanDrive_Socket::SetTimeAsVar(unsigned char VarNumber, TDateTime time)
{
  SetVariable(VarNumber,TimeToUShort(time));
}
//------- -------------------
void TCPU_LanDrive_Socket::SetTimeAsTimer(unsigned char VarNumber, TDateTime
time)
{
  SetTimer(VarNumber,TimeToUShort(time));
}
//------- -------------------
unsigned short TCPU_LanDrive_Socket::TimeToUShort(TDateTime time)
{
   Word Hour, Min, Sec, MSec;
   DecodeTime(time, Hour, Min, Sec, MSec);
   return Hour*100+Min;
}
//------- -------------------
void __fastcall TCPU_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_Socket::GetInternalProtocolVersion(void)
{
char buf[1];
buf[0]='p';
SendBuf(buf,1);
}
//------- -------------------

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

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

void _fastcall TCPU_LanDrive_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_LanDrive_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_LanDrive_Socket.h - бібліотека для файлу CPU_LanDrive_Socket.cpp

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

#ifndef CPU_LanDrive_Socket
#define CPU_LanDrive_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_LanDrive_StateChangeNum {
public:           // Визначено користувачем
unsigned short Num;
void __fastcall Next(void);
__fastcall T_LanDrive_StateChangeNum(void);
};

class T_LanDrive_State {
private:
unsigned char ActiveKey[16];
public:           // Визначено користувачем
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_LanDrive_Socket : public TClientSocket
{
private:
    AnsiString FVersion;
    AnsiString FLogin;
    AnsiString FPassword;
    unsigned char FAuth;
    TOcelotStateChangeNum *OcelotStateChangeNum;
    T_LanDrive_StateChangeNum *_LanDrive_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_LanDrive_Socket(TComponent* Owner);
    __fastcall ~TCPU_LanDrive_Socket(void);
    T_LanDrive_State *_LanDrive_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);
    // Відправка комманд
    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
```

Основна програма

Файл Project_IntellectHome.cpp основної програми

```
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("main.cpp", Form_main); // Головне вікно
USEFORM("about.cpp", Form_about); // Вікно даних про розробника
USEFORM("light.cpp", Form_light); // Вікно управління світлом
USEFORM("water.cpp", Form_water); // Вікно управління водою
USEFORM("gas.cpp", Form_gas); // Вікно управління газом
USEFORM("security.cpp", Form_security); // Вікно управління захистними системами
USEFORM("climate.cpp", Form_climate); // Вікно управління кліматом
USEFORM("phone.cpp", Form_phone); // Вікно управління телефонією
USEFORM("tv.cpp", Form_tv); // Вікно управління телевізорами
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                // Створення головної форми
                Application->CreateForm(__classid(TForm_gas), &Form_gas);
                // Створення форми управління захистними системами
                Application->CreateForm(__classid(TForm_security),
&Form_security);
                // Створення форми управління кліматом
                Application->CreateForm(__classid(TForm_climate),
&Form_climate);
                // Створення форми управління телефонією
                Application->CreateForm(__classid(TForm_phone), &Form_phone);
                // Створення форми управління телевізорами
                Application->CreateForm(__classid(TForm_main), &Form_main);
                // Створення форми даних про розробника
                Application->CreateForm(__classid(TForm_about), &Form_about);
                // Створення форми управління світлом
                Application->CreateForm(__classid(TForm_light), &Form_light);
                // Створення форми управління водою
                Application->CreateForm(__classid(TForm_water), &Form_water);
                // Створення форми управління газом
                Application->CreateForm(__classid(TForm_tv), &Form_tv);
                // Створення форми запуску проекту
                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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:         // Визначено користувачем
        __fastcall TForm_main(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm_main *Form_main;
//---------------------------------------------------------------------
#endif
```

Файл climate.cpp – керування кліматом

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

#include "climate.h"
#include "CPU_LanDrive_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_LanDrive_Command(0,21,18,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
void __fastcall TForm_climate::TrackBar5Change(TObject *Sender)
{
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_StatusQuery();
st=_LanDrive_State->GetStateOnOff(0, 21);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_LanDrive_State->GetStateOnOff(0, 22);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=_LanDrive_State->GetStateOnOff(0, 23);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_LanDrive_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=_LanDrive_State->GetStateOnOff(0, 26);
if(st="On") Label_5->Caption="Відкрито"; else Label_5->Caption="Закрито";
st=_LanDrive_State->GetStateOnOff(0, 27);
if(st="On") Label_6->Caption="Відкрито"; else Label_6->Caption="Закрито";
st=_LanDrive_State->GetStateOnOff(0, 28);
if(st="On") Label_7->Caption="Відкрито"; else Label_7->Caption="Закрито";
st=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __fastcall TForm_climate(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm_climate *Form_climate;
//---------------------------------------------------------------------
#endif
```

Файл light.cpp – керування освітленням

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

#include "light.h"
#include "CPU_LanDrive_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_LanDrive_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_LanDrive_(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_LanDrive_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_LanDrive_Command(0,2,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,3,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,4,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,5,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,6,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,7,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,8,18,1);
```

```
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_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_LanDrive_Command(0,1,18,1);
ClientSocket->SendLeviton_LanDrive_(0,1,TrackBar_torch->Position);

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

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

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

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

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

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


bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
ClientSocket->Send_LanDrive_Command(0,8,18,1);
ClientSocket->SendLeviton_LanDrive_(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_LanDrive_Command(0,1,19,1);

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

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

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

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

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

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


bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
ClientSocket->Send_LanDrive_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_LanDrive_Command(0,x,18,1);
y=random (6)+2;
ClientSocket->Send_LanDrive_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_LanDrive_(0,1,TrackBar_torch->Position);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

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

ClientSocket->Send_LanDrive_StatusQuery();

st=_LanDrive_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=_LanDrive_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=_LanDrive_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=_LanDrive_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=_LanDrive_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=_LanDrive_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=_LanDrive_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=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __fastcall TForm_light(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_light *Form_light;
#endif
```

Файл water.cpp – керування системою водопостачання

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

#include "water.h"
#include "CPU_LanDrive_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_LanDrive_Command(0,9,18,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

//закриття клапану гарячої води на кухні
void __fastcall TForm_water::Button11Click(TObject *Sender)
{
ClientSocket->Send_LanDrive_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_LanDrive_StatusQuery();
st=_LanDrive_State->GetStateOnOff(0, 9);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_LanDrive_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=_LanDrive_State->GetStateOnOff(0, 10);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=_LanDrive_State->GetStateOnOff(0, 11);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";

st=_LanDrive_State->GetStateOnOff(0, 13);
if(st="On") Label_5->Caption="Ввімкнено"; else Label_5->Caption="Вимкнено";
st=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __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_LanDrive_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_LanDrive_Command(0,15,18,1);
}
//---------------------------------------------------------------------------

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

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

void __fastcall TForm_gas::Button3Click(TObject *Sender)
{
ClientSocket->Send_LanDrive_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_LanDrive_StatusQuery();
st=_LanDrive_State->GetStateOnOff(0, 15);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __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_LanDrive_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_LanDrive_(0,32,TrackBar1->Position);
}
//---------------------------------------------------------------------------


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

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

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

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

ClientSocket->Send_LanDrive_StatusQuery();
st=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __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_LanDrive_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_LanDrive_(0,30,TrackBar1->Position);
}
//---------------------------------------------------------------------------

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

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

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

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

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

void __fastcall TForm_phone::Button3Click(TObject *Sender)
{
ClientSocket->Send_LanDrive_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_LanDrive_StatusQuery();
st=_LanDrive_State->GetStateOnOff(0, 30);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";

ClientSocket->Send_LanDrive_StatusQuery();
st=_LanDrive_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:      // Компоненти IDE-управління
        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:    // Визначено користувачем
public:           // Визначено користувачем
        __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_LanDrive_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_LanDrive_Command(0,17,18,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

//визначення та виведення стану системи безпеки
```

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

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

st=_LanDrive_State->GetStateOnOff(0, 19);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_LanDrive_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:      // Компоненти IDE-управління
        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: // Визначено користувачем
public:     // Визначено користувачем
        __fastcall TForm_security(TComponent* Owner);
};
extern PACKAGE TForm_security *Form_security;
#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("БАКАЛАВРСЬКИЙ ПРОЕКТ");
Memo1->Lines->Add("");
Memo1->Lines->Add("на  тему:");
Memo1->Lines->Add("");
Memo1->Lines->Add("Програмне забезпечення системи кібербезпеки для захищеного
управління будинком на основі технології LanDrive");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("Керівник: Смірнова Т.В.");
Memo1->Lines->Add("");
Memo1->Lines->Add("Розробив: студент Царенко Єгор Артурович");
Memo1->Lines->Add("                        гр. КБ-20-3СК");
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:      // Компоненти IDE-управління
        TImage *Image1;
        TMemo *Memo1;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // Визначено користувачем
public:           // Визначено користувачем
        __fastcall TForm_about(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_about *Form_about;
//---------------------------------------------------------------------------
#endif
```