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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Подолян Б.В.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2024 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 *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Подоляну Богдану Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS*

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

затверджені наказом вищого навчального закладу № 132-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS*

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 в промислову експлуатацію.
6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

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

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Подолян Б.В.
(прізвище та ініціали)

АНОТАЦІЯ

Подолян Б.В. Програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

Метою розробки є програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

Результат роботи – програмна реалізація системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, автоматизація будинків, BAS

ABSTRACT

Podolian B.V. Software of the system of intelligent home automation services using BAS technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the system of intelligent home automation services using BAS technology.

The goal of the development is the software system of intelligent home automation services using BAS technology.

The result of the work is the software implementation of a system of intelligent building automation services using BAS 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 a PC with Windows 10/11 OS.

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

Keywords: computer engineering, home automation, BAS

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. Системи управління будівлями (BMS) часто називають кількома різними назвами, включаючи «системи автоматизації будівель (BAS), «системи управління будівлями» і навіть «розумні будівлі».

Незалежно від термінології, яку ви віддаєте перевагу, вона складається з розподіленої системи керування, яка об’єднує різні типи будівельних систем в одному централізованому місці. Ці системи керування будівлею контролюють опалення, вентиляцію та кондиціонування повітря (HVAC).

BMS працює як система комп’ютерної мережі, яка відстежує та контролює ряд інших електронних і механічних систем. Ці системи можуть спілкуватися між багатьма платформами, програмним забезпеченням і мовами.

BMS включає більше, ніж просто один даховий блок HVAC, вентилятори або інструменти для очищення. Визначальною особливістю є те, що він об’єднує всі механічні та електронні системи на одній панелі керування для централізованого керування.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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

– Огляд існуючих систем інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

– Дослідження системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

– Програмна реалізація системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS, 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

У базі «розумних» будинків та інтелектуальних будинків майбутнього – конвергентна структурована кабельна мережа. Конвергенція забезпечує цілий ряд переваг і дозволяє використовувати для реалізації широкого спектра інтелектуальних сервісів систем автоматизації будинків (Building Automation System, BAS) одну структуровану кабельну мережу. Сервіси керування енергоспоживанням, освітленням, безпекою, цифровими вивісками, системами опалення, вентиляції й кондиціювання (HVAC) можуть надаватися на основі тої ж самої мережної інфраструктури, що служить для підключення бездротових точок доступу, а також забезпечує підтримку передачі голосу й даних.

Застосування для цих цілей однієї структурованої кабельної мережі дозволяє значно скоротити кількість кабельних каналів і кабелів різного типу. Подальшому зменшенню їхнього числа сприяють такі стратегії, як Cable Sharing, коли трохи низькошвидкістних додатків високої щільності розгортаються з використанням одного каналу Категорії 7_A / Класу F_A. Крім того, можливість підтримки технологій віддаленого електроживлення, таких як Type 1 PoEwer over Ethernet (PoE) на 15 Вт і Type 2 PoE на 30 Вт, означає, що єдина конвергентна мережа виключає необхідність установки зовнішніх розеток живлення біля слабкострумових пристроїв BAS.

1.2 Область застосування

Типи систем управління будівлями

Розуміння різних типів систем BMS має важливе значення для прийняття більш обґрунтованого рішення. Ось розбивка:

Керування опаленням, вентиляцією та кондиціонуванням повітря (HVAC) – керує опаленням, охолодженням і вентиляцією в будівлі. Орендарі будуть насолоджуватися комфортними умовами та хорошою якістю повітря.

Керування освітленням – регулює освітлення в будівлі за допомогою датчиків і таймерів для регулювання освітлення. Працює залежно від зайнятості кімнати або часу доби.

Безпека та контроль доступу – ці системи контролюють доступ до будівлі та запобігають несанкціонованому проникненню. Стандартні функції включають ключ-картку доступу, відеоспостереження та системи сигналізації.

Системи енергоменеджменту будівлі (EMS) – відстежує та контролює споживання енергії будівлею. Ця система спрямована на зниження витрат і підвищення енергоефективності.

Системи пожежної сигналізації та безпеки – ці критично важливі системи активують сигналізацію під час пожежі та негайно повідомляють екстрені служби.

Система управління та контролю будівлі (BMCS) – більш комплексна система, яка об'єднує кілька із зазначених вище систем (наприклад, HVAC, освітлення та безпека) в єдиний інтерфейс. Це дозволяє більш ефективно та централізовано керувати функціями будівлі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS, 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

AMX

Корпорація AMX (Даллас, США) – один з найбільших розроблювачів систем інтелектуального керування встаткуванням. На сьогоднішній день технологічні рішення AMX представлені більшістю компаній, що спеціалізуються на установці встаткування для розумного будинку. У числі плюсів пропонованих компанією пропозицій можна відзначити модульний підхід, що дозволяє гнучко підходити до адаптації проектів під потреби замовника. Відразу відзначимо, що ціни цієї компанії можна віднести до верхнього цінового сегмента. Установка й монтаж комплексного рішення на базі встаткування AMX обійдеться від 20 до 120 тис.доларів.

WeMo

Строго говорячи WeMo від компанії belkin – це прилади для керування електроприладами, а не комплексне рішення для "розумного" будинку. Однак, з огляду на доступну вартість, вона може стати першим кроком на шляху автоматизації житла. У лінійку продуктів WeMo входять: WeMo Home Control Switch – керування електропристроями, які приєднуються до електромережі через цей апарат. З його допомогою можна настроїти режим включення/вимикання, а також управляти пристроями віддалено, через хмарні сервіси, повідомлення SMS і голосові команди.

WeMo motion Sensor – керування освітленням також, за допомогою настроювання режимів включення/вимикання, а також за допомогою команд, які поступають від убудованих датчиків руху.

WeMo BabyMonitor – призначений для виявлення й передачі інформації про виявлення звуків на мобільні пристрої.

Вартість одного комплекту – від 100 до 200 доларів.

Control4

Control4 – канадська компанія, що пропонує технологічне рішення для автоматизації житла, яке можна віднести до середнього цінового сегмента. На цьому ринку бренд досить відомий і авторитетний. Одними з перших комплексно сталі використовувати у своїх рішеннях бездротові протоколи передачі даних, за рахунок чого забезпечили значне зниження вартості.

Залежно від функціонала, ціна установки може варіюватися від 1,5 тис.дол для автоматизації домашнього кінотеатру, до 100 тис дол для комплексної автоматизації котеджу (освітлення, штори, жалюзі, аудіо й відео мультирум, домашні кінотеатри, захист від протікань, клімат-контроль, домофонія, керування встаткуванням забезпечується за допомогою Apple iPad і із сенсорних панелей 7" і 5", аудіо й відео інтерком).

Crestron

Crestron – одна із самих досвідчених компаній, що займаються автоматизацією домашнього й офісного встаткування. Пропонує як модульні, так і комплексні рішення. Одне з переваг системи – можливість настроювання сценаріїв самим користувачем, хоча, у цьому зв'язку, накладаються обмеження на функціональність. Створюється враження, що більшою мірою компанія профілюється на рішеннях для офісів і організацій, оскільки спектр пропозицій у цій частині в компанії більше багатий, ніж аналогічні рішення конкурентів. Також, хтось може віднести до числа переваг можливість інтеграції встаткування від Crestron із пристроями на базі iOS. Розцінки порівнянні із пропозиціями від Control4.

EnOcean

EnOcean – це набір комплектів для установки окремих функцій "розумного" будинку. Може бути дуже вдалим рішенням для тих, хто не

прагнути встановлювати комплексні рішення. Все встаткування бездротове й не вимагає джерел живлення, оскільки бере енергію від навколишнього середовища (світло, перепад температур, тиск і т.д.).

Можливе придбання комплектів для реалізації наступних функцій:

– Включення освітлення при виявленні руху.

– Плавне збільшення/зменшення яскравості штучного освітлення.

– Керування нагрівальними приладами на підставі дані температури приміщення.

– Керування електроприладами за допомогою дистанційного пульта.

– Включення/вимикання штучного освітлення на підставі яскравості природного освітлення.

Кожний комплект устаткування EnOcean (вимикач, датчик, панель керування) коштує від 5000 до 15000 грн, що дозволить впровадити елементи автоматизації в себе будинку за невеликі гроші.

Home Sapiens

Home Sapiens – це програмне забезпечення, що дозволяє настроїти керування домашнім устаткуванням за допомогою голосу, комп'ютера й сенсорних пристроїв (смартфон, планшет). Підтримує найбільш популярні провідні й бездротові протоколи передачі даних – Z-Wave, X10, KNX, Smart-Bus.

Вартість розумного будинку на базі Home Sapiensможет обійтися від 15000 грн за базову комплектацію (програмне забезпечення й мінімальне встаткування). Вартість ліцензії тільки програмного забезпечення становить порядку 2000 грн. Це дасть можливість управляти технікою за допомогою голосу, віддалено через телефон, набудовувати будильник і інформаційну систему.

При бажанні можна обладнати "розумний" будинок сенсорними панелями, охоронною сигналізація з датчиками й т.д. У цьому випадку вартість на базі Home Sapiens виросте до 400000 грн. (10000 доларів).

HomeSeer

HomeSeer, також, є програмним забезпеченням, з підтримкою великого списку протоколів провідної й бездротової передачі даних – Insteon, UPB, X10, xAP, xPL і Z-Wave і інші. У цей час компанією випущений контролер на базі Z-Wave, що дозволяє інтегрувати обладнання в кероване середовище "розумного" будинку.

Комплексна автоматизація на базі цієї технології обійдеться від 150000 грн (близько 5000 доларів) за 1-2 комн. квартиру.

Insteon

INSTEON – це технологія організації й передачі даних для керування встаткуванням інтелектуального житла. Технологія розроблена порівняно недавно. При її створенні були враховані недоліки колишніх протоколів передачі даних, призначених для автоматизації домашнього встаткування, і, за рахунок рятування від зайвого функціонала, удалося оптимізувати систему й забезпечити доступну ціну.

Кожний елемент системи INSTEON (контролери, пульти, датчики й т.п.) обійдеться Вам у трохи тис.грн. (від 2000 до 8000)

KNX

Стандарт KNX на сьогоднішній день активно застосовується в багатьох офісах і будинках. Подальша тенденція розвитку технології KNX напевно зробить її ще більш популярної, оскільки вже зараз основні німецькі виробники стандарту орієнтуються на забезпечення їм звичайних будинків і квартир, з максимально зручним контролем освітлення, температурним режимом і іншими побутовими параметрами. Крім них, технології KNX дозволяють управляти всіма іншими функціями будинку – охоронною сигналізація й контролем допуску в будинок, вентилюванням і кондиціюванням повітря, обігрівом будинку й забезпечення його водою. Вартість рішення на базі KNX може становити від 500000 грн за 1 кмн. кв до 2000000 грн за котедж (у доларах – від 15000 до 60000)

Loxone

Loxone представлений на ринку пристроїв для автоматизації житла мінісервером, що дозволяє забезпечувати контроль устаткування. Дозволяє інтегруватися з більшістю відомих пристроїв, включаючи смартфони й планшети, забезпечуючи, таким чином, гнучкість установки й настроювання встаткування.

Сам сервер Loxone коштує від 25000 грн. Комплексне рішення від 120 до 300 тис.грн (від 3,5 до 10 тис. доларів)

Mi Casa Verde

Mi Casa Verde – компанія з Гоконга, що робить контролери для автоматизації житла «Vera» на операційній системі Linux. Основним протоколом компанія вибрала Z-Wave, але також підтримуються X10, Control4, Isteon.

Вартість пропозиції для квартири на базі Mi Casa Verde може становити від 100000 грн (3 тис. доларів)

X10

X10 – сама заслужена й відома система автоматизації житла. Величезна кількість устаткування для розумного будинку підтримує саме цю технологію. Можливість вибору дозволяє знайти комплектуючі за невисокою ціною. Приміром, мінімальний комплект для автоматизації освітлення може скласти 5000 грн. На сьогоднішній день технологія вважається застарілої через невисоку швидкість передачі даних і відсутності зворотного зв'язка із пристроями.

Z-Wave

Z-Wave – це бездротова технологія, що забезпечує комунікації між пристроями за рахунок самих пристроїв. Інакше кажучи, кожний елемент системи є й приймачем, і передавачем сигналу (так звана ніздрювата мережа). Це дозволяє забезпечити високу надійність системи, що працює навіть у випадку виходу з ладу окремих її елементів. Цей підхід дозволяє гнучко підходити до зоні покриття системи, що не бідує в додаткових підсилювачах сигналу у випадку збільшення площі охвату. Вартість пропозиції для квартири на базі Z-Wave може становити від 100000 грн.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними

під час роботи програми, тому вносити зміни можна в процесі відлагодження.

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

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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 контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

В Україні вже давно прижилися й улаштувалися в більшості середньостатистичних будинків досить примітивні, малофункціональні й не особливо привабливі зовні домофони, смутні відео-домофони із чорно-білою, нечіткою й перекрученою картинкою. Прогрес не стоїть на місці й у даній сфері, а тому всі частіше цим пристроям протиставляють сучасні й багатофункціональні IP-домофони BAS-IP ltd, які легко інтегруються в популярну в усьому світі систему "Розумний будинок" і дозволяють створювати її з нуля.

Перші аудіо-відеодомофони були запропоновані для того, щоб створити хоча б незначну перешкоду на шляху кримінальних і небажаних елементів. Зловмисників такі пристрої пригальмували ненадовго. Вони швидко освоїлися.

У наші дні для захисту квартир, під'їздів, приватних будинків, офісів від несанкціонованого проникнення пропонується якісне відео-устаткування, створене на базі інформаційних технологій. З його допомогою виробляється не тільки віддалене керування вхідними дверима, але й безліч інших функцій.

Останні інноваційні рішення, які назвали IP-домофонами, наділені безліччю унікальних можливостей. У складі комплексу "Розумний будинок" вони дозволяють віддалено управляти:

– освітленням;

– мікрокліматом;

– піднімальними пристроями й т.д.

Серію домофонів такого інноваційного типу пропонує виробничий бренд British Advanced Solutions Ltd. Марка відома на українському й світовому ринках, як BAS-IP. З 2008 року вона робить спеціалізовані пристрої для побудови систем доступу й "Розумного будинку".

BAS-IP працює вже давно, просто раніше вона була частиною телекомунікаційної структури. Історичне минуле дозволило фахівцям одержати серйозний досвід і загартування в сфері інтеграції IP-технологій, у тому числі, взаємодії цифрової продукції з аналоговим устаткуванням.

У наше століття, коли про системи із цифровою передачею даних так багато говорять, розробки BAS-IP довелися, як не можна до речі. У гранично короткий термін марка зуміла влаштуватися на ринку інформаційного встаткування. Залучає споживача й той факт, що всі вироби споконвічно реалізуються по доступній для такого рівня встаткування ціні.

Виробу під маркою BAS-IP завжди відповідає міжнародним і українським нормам. Вона сертифікована. Досягнення компанії в сфері виробництва встаткування на базі інформаційних технологій багато разів відзначалися нагородами.

Особливості модельних ліній BAS-IP

Виробу BAS-IP будуються на широких можливостях одночасно IP-відеоспостереження й VoIP, тобто IP-телефонії. Компанія пропонує:

– IP-системи для встаткування домофонів;

– монітори, що мають сучасні тачпади;

– антивандальні викличні панелі;

– модулі керування;

– СКУД.

Виробництво IP-домофонів є пріоритетним для BAS-IP. Завдяки посиленим технологічним розробкам і сучасній організації виробничих потужностей, бренду вдалося оперативно запустити потокові лінії й налагодити серійне виробництво зовсім унікальних модельних ліній.

Все встаткування BAS-IP відповідає актуальним вимогам користувача. Приємно й те, що особлива увага була приділена стилю корпусів всіх виробів. Дизайн порадував стомленого типовими пристроями споживача. В елегантних і

компактних корпусах, виконаних із сучасних матеріалів, ховається найбагатший функціонал, а керування пристроями гранично спрощене й інтуїтивно зрозуміло.

Для всіх внутрішніх моніторів пропонується сенсорний варіант керування, а зовнішні панелі продумані з антивандальної точки зору. Їхня надійність перевірена на сучасних стендах. Тести й випробування наочно продемонстрували стійкість виробів до механічних впливів.

Поставки встаткування BAS-IP у Україну були налагоджені й спростилися після того, як в 2011 році компанія відкрила філію в столиці. IP-домофони виробника сьогодні легко інтегруються в існуючі мережі, установлені локально:

– у квартирах і багатоквартирних будовах,

– цілих заміських селищах і окремих заміських котеджах,

– офісних центрах і невеликих офісах.

При організації контролю доступу за допомогою товарів марки BAS-IP інформація від компонента до компонента, від пристрою до пристрою в створених системах передається по протоколах TCP/IP. Застосування протоколів SIP і керованість всіма елементами за допомогою коммуникаторів, смартфонів на Android або iOS робить IP-домофони й системи виробника особливо популярними.

Застосування сучасних протоколів дозволяє забезпечити самий безпечний канал для взаємодії користувачів і тих, хто намагається одержати доступ до внутрішнього простору охоронюваної території. Пристрою BAS-IP наділені цілим рядом додаткових можливостей, що визначило із застосування при створенні "Розумного будинку". Пристрою дозволяють:

– здійснювати якісну з консьєржем або абонентом (аудіо-, відео-);

– управляти всієї "Розумним будинком";

– підключати відеокамери й так далі.

Сучасні модельні лінії BAS-IP підключаються за допомогою Ethernet (PoE), так що питання живлення спростилося й не потрібна прокладка додаткового кабелю.

Пристрої BAS-IP для побудови "Розумного будинку"

Для пристрою "Розумний будинок" пропонує:

– SH-61 – конвертер протоколу, що перетворює команди, що надходять із монітора усередині охоронюваних територій у сигнали, що дозволяють управляти кінцевими автоматичними модулями;

– SH-62 – модуль керування чотирма окремими каналами організації освітлення приміщень (тригерний тип роботи), управляє включенням/відключенням з можливістю підключення до монітора до восьми модулів (у цілому – 32 каналу на чотири приміщення);

– SH-63 – модуль для організації керування двома занавесями на відкривання/сакривання з паузою в різних положеннях (виводиться до двох модулів на монітор, чотири канали, по одному на приміщення);

– SH-64 – модуль керування двома кондиціонерами з можливістю включення системи охолодження/нагрівання, старту вентилятора, керування режимами (на виводиться до двох модулів на монітор, чотири канали, по одному на приміщення);

– EVRC-16 – модуль керування блоком центрального контролера піднімальних пристроїв з функцією виклику на поверхи (до 16-ти) або з'єднанням кнопок виклику з монтажною колодкою;

– SH-67 – програмувальний інфрачервоний модуль керування ІЧ-устаткуванням (ТВ, DVD, кондиціонер) через ІЧ-порт. BAS-IP пропонує зовсім інше рішення для створення систем контролю доступу з домофонами.

Найважливіша продукція в модельних лініях BAS-IP – це, звичайно ж, домофони. Вони:

– прості в монтажі й керуванні;

– пропонують широку функціональність, сучасний ергономічний дизайн;

– виконують величезне число функцій, відсутніх в аналогових домофонах і пристроях, включаючи забезпечення повної безпеки приміщень;

– легко піддаються модернізації й дозволяють розширювати систему;

– спрощують роботу з кабелем у системі й заощаджують витрати на ньому, завдяки використанню архітектури TCP/IP.

Якщо який-небудь вузол виходить із ладу, не важко буде замінити його, не міняючи систему в цілому. Монтаж домофоних систем BAS-IP може виконати фахівець із локальних мереж. Для цього не буде потрібно підвищувати кваліфікацію або проходити навчання.

Що таке IP-домофон BAS-IP Продукція британської компанії дає можливість одержувати ідеальну картинку з відеокамери викличної панелі. Передача інформації ведеться з усуненням навіть найменших шумів і перешкод на лінії. Підключення домофона ми здійснюємо за допомогою кручений пари, оскільки на пристроях передбачене рознімання RJ45.

Лінійка встаткування BAS-IP дозволить:

– використовувати вже існуючу в будинку або квартирі локальну мережу, включаючи кабель для Інтернет-коннекта, і не зажадає прокладки додаткових кабелів;

– санкціоновано відкривати різні двері, хвіртки, ворота в будинках, будовах і прилягаючих територіях відповідно до наданого рівня доступу;

– переглядати зображення з IP-відеокамер, що є присутнім у мережі;

– одержувати інформацію з викличних панелей відповідно до доступів;

– спілкуватися з консьєржем за допомогою всього однієї кнопки керування.

Мультифункціональні монітори із сенсорним керуванням прийдуть по смаку всім, хто віддає перевагу високому й сучасному рівню комфорту й звичний спосіб керування гаджетами й девайсами. Ми пропонуємо встановити індивідуальні й багатоквартирні відеодомофони BAS-IP при встаткуванні системи "Розумний будинок" і не тільки, щоб:

– користуватися максимальним комфортом і простотою індивідуальних настроювань устаткування;

– одержати можливість монтажу будь-якого числа додаткових IP-відеокамер і використовувати пристрій при створенні ефективної системи відеоспостереження, забезпечення контролю доступу на охоронюваних територіях;

– користуватися зручною й сучасною функцією аудіо- і відеоінтеркома, щоб оперативно вести переговори з абонентами, що перебувають на більших охоронюваних територіях;

– підключати флеш-карти для копіювання й збереження отриманої картинки у форматі відео й фото;

– відкривати замки;

– викликати піднімальні пристрої (ліфти);

– контролювати освітлення на власній території;

– управляти підключеною домашньою електронікою й автоматикою;

– організувати загальну систему контролю за допомогою установки датчиків руху, задимлення, витоку води, газу;

– використовувати додатковий режим фоторамки для перегляду фотографій;

– дистанційно управляти всіма функціями за допомогою передвстановленого й доданого ПЗ на базі Android і iOS.

Докладніше про різні підходи до зонування, включаючи кругові, стільникові й прямокутні сітки, розповідається в документах EN 50173-6 Distributed Building Services, ANSI/ TIA-862-A Building Automation Systems Cabling Standard і TIA TSB-162-A Telecommunications Guidelines for Wireless Access PoEints. Помітимо, що термінологія в стандартах EN і TIA розрізняється: в TIA-862-A говориться про розетки для підключення встаткування (Equipment Outlet, EO) і горизонтальних точках підключення (Horizontal Connection PoEint, HCP), а в BS EN 50173-6 – про сервісні розетки (Service Outlet, SO) і сервісних точках підключення (Service Connection PoEint, SCP).

3.2 Розробка структурної схеми

При організації бездротового доступу архітектори мереж повинні враховувати тенденцію до повсюдного поширення концепції BYOD, очікування збільшення швидкостей передачі даних для підтримки потокового відео й мультимедіа, а також тарифні плани операторів мобільних мереж, що сприяють розвантаженню трафіку за рахунок мереж Wi-Fi. Оскільки Wi-Fi стає пріоритетним способом доступу для пристроїв BYOD, більше «швидке» устаткування для бездротових локальних мереж буде сприяти запобіганню появи вузьких місць і перевантажень, збільшенню пропускної здатності й зниженню затримок, однак упоратися с цими завданнями воно зможе лише в тому випадку, якщо кабельна інфраструктура й сполучне встаткування зможуть підтримувати зрослу пропускну здатність.

Гарна новина полягає в тому, що зонна топологія, що рекомендується для конвергентних мереж, відмінно підходить для установки точок бездротового доступу (Wireless Access PoEint, WAP) – з урахуванням всіх рекомендацій і вимог до покриття мережі. Однак критично важливий і вибір СКС. У цей час для підключення пристроїв 802.11ac із трьома просторовими потоками й підтримки швидкості 1,3 Гбіт/с потрібно два фізичних з'єднання 1000 Base-T, тобто агрегування каналів, як показано на рисунку 3.1.

З урахуванням майбутніх удосконалень WAP – потенційної швидкості 2,6 Гбіт/с (три потоки 160 МГц) і інших реалізацій 802.11ac – для кожної точки бездротового доступу рекомендується використовувати дві лінії Класу EA / Категорії 6A або вище. Крім того, WAP 802.11ac повинна мати адаптер живлення DC або одержувати живлення по мережі Type 2, іноді неформально називане PoE Plus. Для підключення WAP 802.11ac рекомендується екранована проводка, що добре розсіює тепло й яка зберігає свої характеристики при нагріванні кабелю.

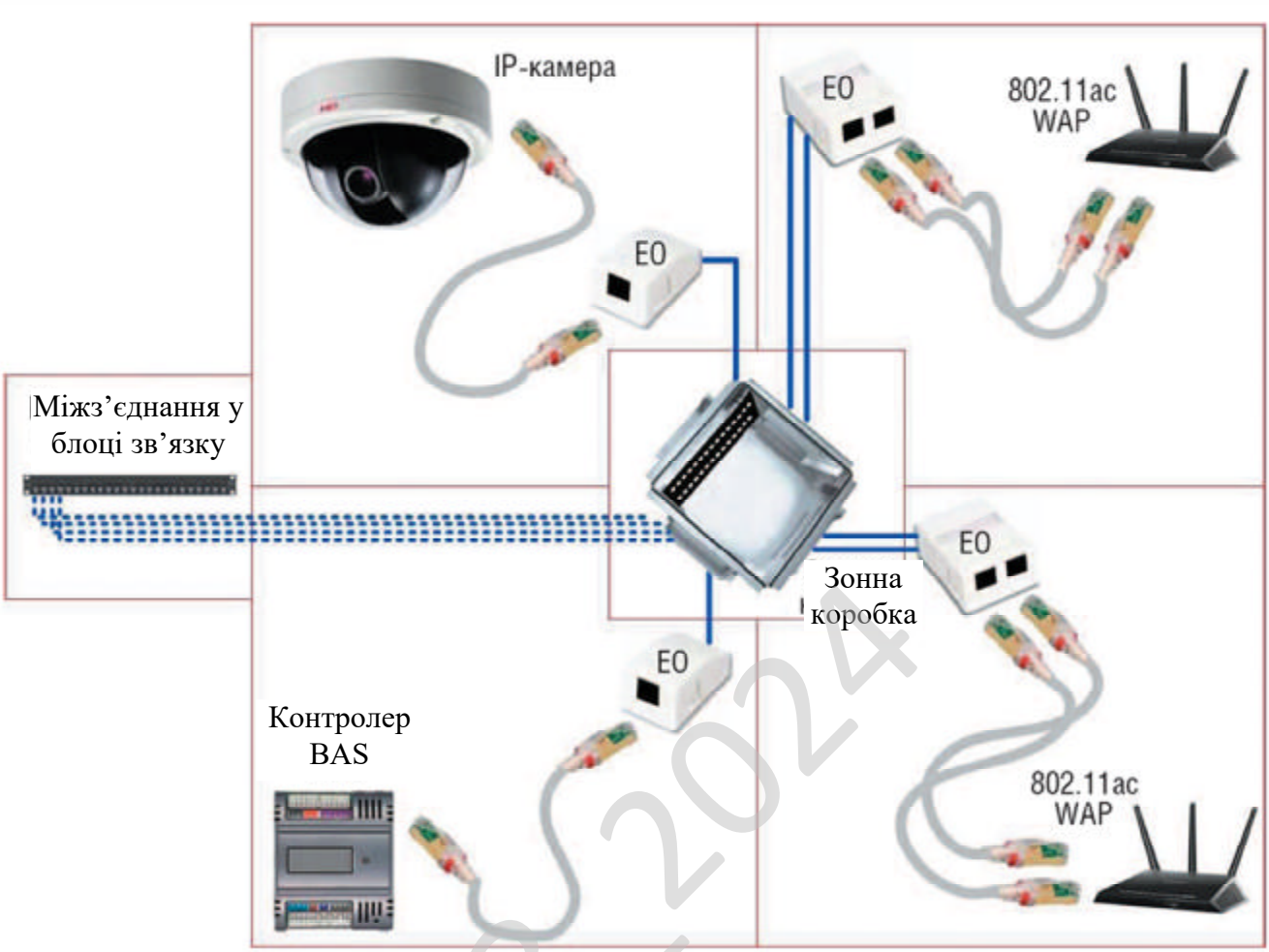

Рисунок 3.1 – Структурна схема системи

При проектуванні конвергентної кабельної інфраструктури, для надійного розгортання точок доступу 802.11ac потрібно брати до уваги й розповсюджені в цей час швидкості передачі даних при з'єднаннях з комутаторами, серверами й пристроями, а також стратегії забезпечення надмірності, відновлення встаткування й майбутні бездротові технології. Для організації мережі 802.11ac у діапазоні 5 ГГц потрібно відносно щільне покриття площ точками доступу WAP. Ідеальний спосіб забезпечити достатнє число резервних портів для агрегування каналів 1000 Base-T до кожної точки доступу 802.11ac – побудова сітки з екранованої СКС Категорії 6А з використанням точок консолідації в зонних

коробках. З появою на ринку встаткування Wi-Fi з портами10 GBase-T це дозволить ефективно їх використовувати.

Екрановані кабельні системи Класу E_A / Категорії 6A і Класу F_A / Категорії 7_A, що забезпечують механічну стабільність до 75°C, дозволяють використовувати більше довгі канали при підвищеній температурі, інакше для відповідності показника внесених втрат вимогам TIA і ISO/IEC максимальну довжину каналу прийде обмежити. Такі системи рекомендуються для додатків з живленням потужністю до 30 Вт і вище, а також у місцях, де температура навколишнього середовища виходить за межі 20°C. Крім того, у пучок можна укладати більше число кабелів без побоювань перегріву.

Причини привабливості конвергентної мережі IP/Ethernet цілком очевидні. Самим розумним буде сполучення гнучкості й довговічності, що дозволяє враховувати майбутні потреби в продуктивності. У цей час СКС Категорії 7_A / Класу F_A мають найвищі для кручений пари характеристики, що забезпечує максимальну пропускну здатність для всіх конвергентних додатків, включаючи вимоги 10 GBase-T і віддаленого електроживлення PoE Type 2 для революційного додатка 802.11ac Wi-Fi. Уперше специфікації високопродуктивної кабельної проводки Класу E_A / Категорії 6A і вище стають для конвергентних мереж критично важливими, що дозволяють підтримувати з'єднання з комутаторами й точками доступу нового покоління для досягнення мультигігабітной пропускної здатності.

Автоматизація будівель та енергоменеджмент

Більше половини великих будівель (використовують BMS.

Дослідження показали , що за останні десять років системи автоматизації будівель продемонстрували низку енергозбереження. Цей діапазон коливається від нуля до понад 30%.

За нашими оцінками, економія коштів буде ще значною для старих будівель або тих, які потребують кращого обслуговування.

Чим корисні системи автоматизації будівель?

Добре працююча система автоматизації будівлі має багато переваг, таких як:

– Розширений BAS може одночасно керувати багатьма системами (такими як захист від пожежі та повеней, вентиляція та охорона) і значно зменшити ймовірність людської помилки.

– Можливість моніторингу роботи різних систем (опалення, повітря, освітлення та ін.).

– Забезпечте безвідмовні механізми для підключення до мережі в разі виникнення електронних або механічних збоїв. Це особливо важливо в небезпечних умовах роботи з високим рівнем ризику.

– Підвищення ефективності інших систем у будівлі.

– Система енергоменеджменту може зменшити споживання енергії та експлуатаційні витрати різних систем.

– Подовжте термін служби різних інженерних комунікацій, що призведе до менш частих ремонтів.

– Забезпечте постійний рівень комфорту для мешканців.

– Може здійснювати блокування, щоб гарантувати, що обладнання вмикається лише тоді, коли це потрібно.

– Виконайте діагностику для моніторингу температури, тиску, витрати тощо в різних системах.

– Добре інтегрований BAS усуває резервування , яке може виникнути, коли занадто велика частина автоматизації в будівлі перекривається.

– Система управління будівлею оптимізує технічне обслуговування, стримуючи витрати, прогнозуючи проблеми до їх загострення. Він також підвищує безпеку за допомогою інтегрованого спостереження та контролю доступу, забезпечуючи безпечніше середовище.

Які деякі системи контролює BAS?

Ви можете не знати про численні системи, якими керує BAS. Ось кілька прикладів:

– **HVAC** – Системи управління будівлею працюють з найбільш енергоефективними системами HVAC. Це найпоширеніший спосіб інтеграції BAS в електронні та механічні системи будівлі.

– **Освітлення** – може керувати освітленням у кімнаті залежно від фактичної кількості людей. Він також може керувати освітленням на основі денного та нічного світла.

– **Система безпеки** – Системи управління будівлею можуть визначати, коли вмикати та вимикати системи безпеки. Вони керують відеоспостереженням, безключовим входом, паролем і віддаленим доступом.

– **Системи вентиляції/очищення повітря** – BMS керує вентиляцією та очищенням повітря, особливо в будівлях, де це має вирішальне значення. Ця система зменшує потребу в нагляді з боку людини та прийняття рішень на місці.

– **Пристрої обробки повітря** – система BAS необхідна для обслуговування пристроїв обробки повітря на нафтопереробних заводах, хімічних заводах або атомних електростанціях. Механізовані повітрозабірники мають необхідні засоби захисту та подвійне посилення для забезпечення безпеки працівників і населення на території.

– **Системи стерилізації/дезінфекції** – Системи автоматизації будівель необхідні для належного функціонування стерилізаційних установок.

– **Стійке середовище** – система управління будівлею необхідна для стабільного функціонування стійкого середовища, наприклад для моніторингу життєво важливих функцій людини або тварин. У таких випадках найкраще мати повністю автоматизований процес керування середовищем.

Як BAS може вам допомогти

Системи автоматизації будівель непомітно працюють у фоновому режимі. Якщо ви їх помітили, вони не виконують свою роботу. Отже, як вони вам допомагають?

– **Для менеджерів об'єктів** – Системи керування будівлями можуть полегшити управління об'єктом (незалежно від типу об'єкта). Така система може допомогти керівникам об'єктів краще керувати втратами енергії та неефективністю.

– **Для власників бізнесу** – автоматизація будівель може скоротити витрати на електроенергію, подовжити термін служби обладнання (наприклад, систем опалення, вентиляції та кондиціонування) і підвищити комфорт орендарів.

– **Для керівників лікарень або операційних директорів лікарень** – комплексна, добре функціонуюча та актуальна BAS є важливою. Лікарня може являти собою одну будівлю або низку будівель, які повинні працювати ефективно в будь-який час. Система опалення, вентиляції, кондиціонування повітря та освітлення – це кілька загальних вимог. Інші не менш важливі – вентиляція, системи стерилізації, механізовані протоколи санітарії та очищення повітря.

– **Для менеджерів нерухомості та власників будівель** – насолоджуйтесь миттєвим доступом до єдиної інформаційної панелі, до якої входять усі інші автоматизовані системи. Таким чином ви можете керувати освітленням, опаленням, вентиляцією, кондиціонуванням, ліфтами, охороною тощо з одного місця.

Як встановити та підтримувати систему BMS

Ми настійно рекомендуємо співпрацювати з професіоналами, які спеціалізуються на встановленні BMS . Самостійно встановлювати систему з метою економії не варто. Відсутність досвіду та глибокого розуміння призведе до множинної неефективності, яка зведе нанівець вигоди, які ви сподіваєтеся отримати.

МАСС забезпечить ретельний процес встановлення та відповідність конкретним вимогам вашої будівлі. Ми проаналізуємо поточні системи будівлі, створимо відповідний проект, проведемо необхідну електропроводку та налаштуємо програмне забезпечення.

Дуже важливо переконатися, що нова система керування будівлею може працювати з усіма існуючими. Встановлення професіоналами дозволяє новій системі добре взаємодіяти з існуючою та запобігає втраті функцій.

Регулярне профілактичне обслуговування BAS також є обов'язковим. Періодичні перевірки, випробування та калібрування допомагають забезпечити довший термін служби та менше проблем. Оновлення програмного забезпечення за необхідності та заміна зношених компонентів забезпечують безперебійну та ефективну роботу BAS.

Системи автоматизації будівель – це найкращий шлях

Добре інтегрована система BMS забезпечує покращений комфорт орендарів, енергоефективність та економічну ефективність операцій. Інвестування у високоякісну систему управління будівлею – це розумний крок із довгостроковими перевагами для вашої організації.

Завдяки покращеному контролю над системами вашої будівлі ви можете розраховувати на значну рентабельність інвестицій.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Функціональна структура розробленої мережі складається з наступних блоків, як взаємодіють з блоком функцій BAS:

– 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 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.і.);

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

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

– управління з екрану ЕОМ;

– релейне управління;

– діммування;

– «проходні» схеми вимикання;

– бездротове управління електрикою;

– для аудіо/відео та світла один пульт ДУ;

– віддалене управління за телефоном.

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 роботу з мультимедіа, включає в себе наступні функціональні блоки:

– бездротове аудіо/відео;

– універсальні навчаємі пульти ІЧ ДУ;

– радіотрансляція ІЧ ДУ;

– бездротове аудіо/відео спостереження;

– радіоуправління ПЗ ЕОМ.

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

– звукова тривожна сигналізація;

– охоронна сигналізація;

– світлова тривожна сигналізація;

– тривожне оповіщення по телефону;

– світлова імітація наявності людей;

– функція «Паніка».

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

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

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

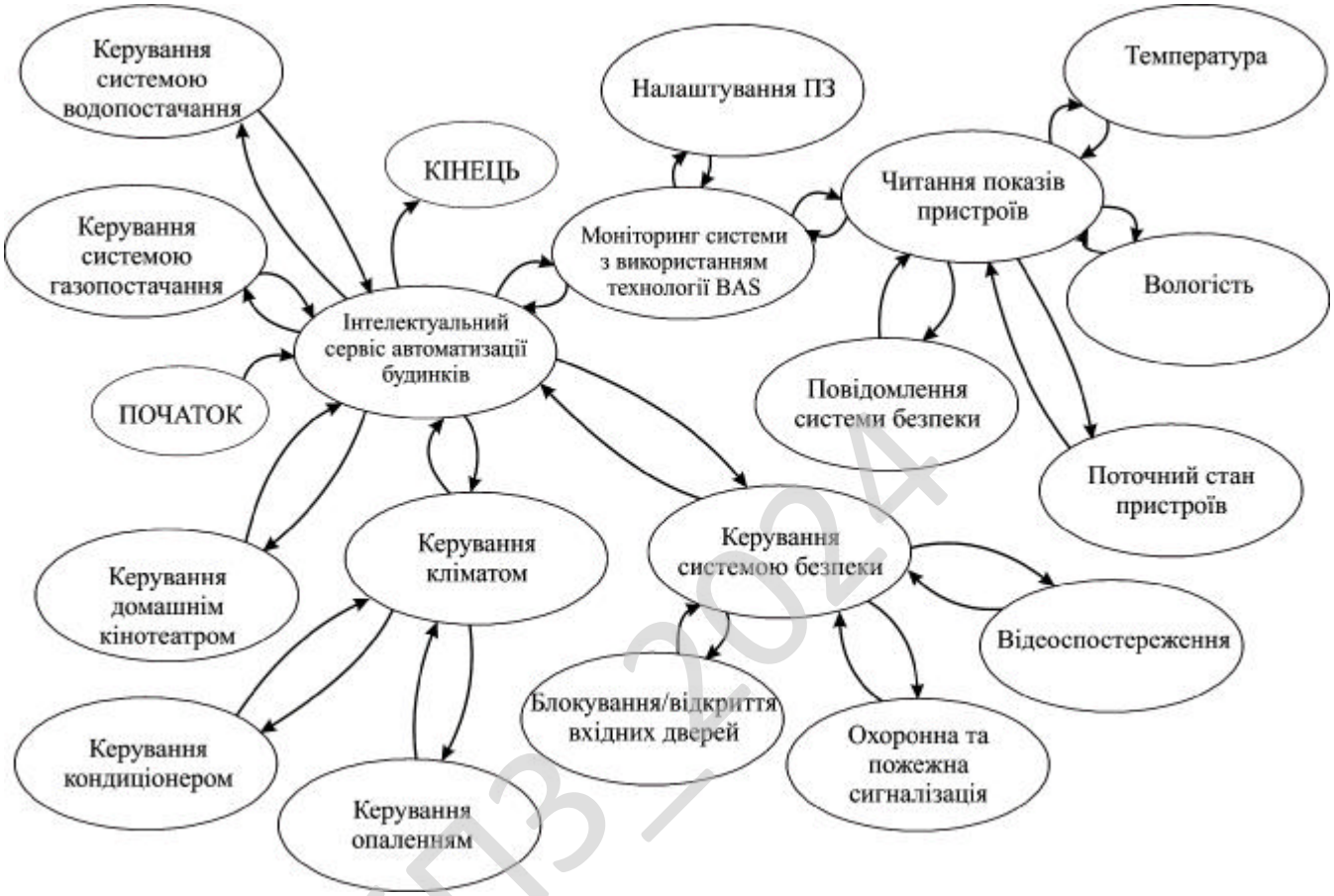

Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

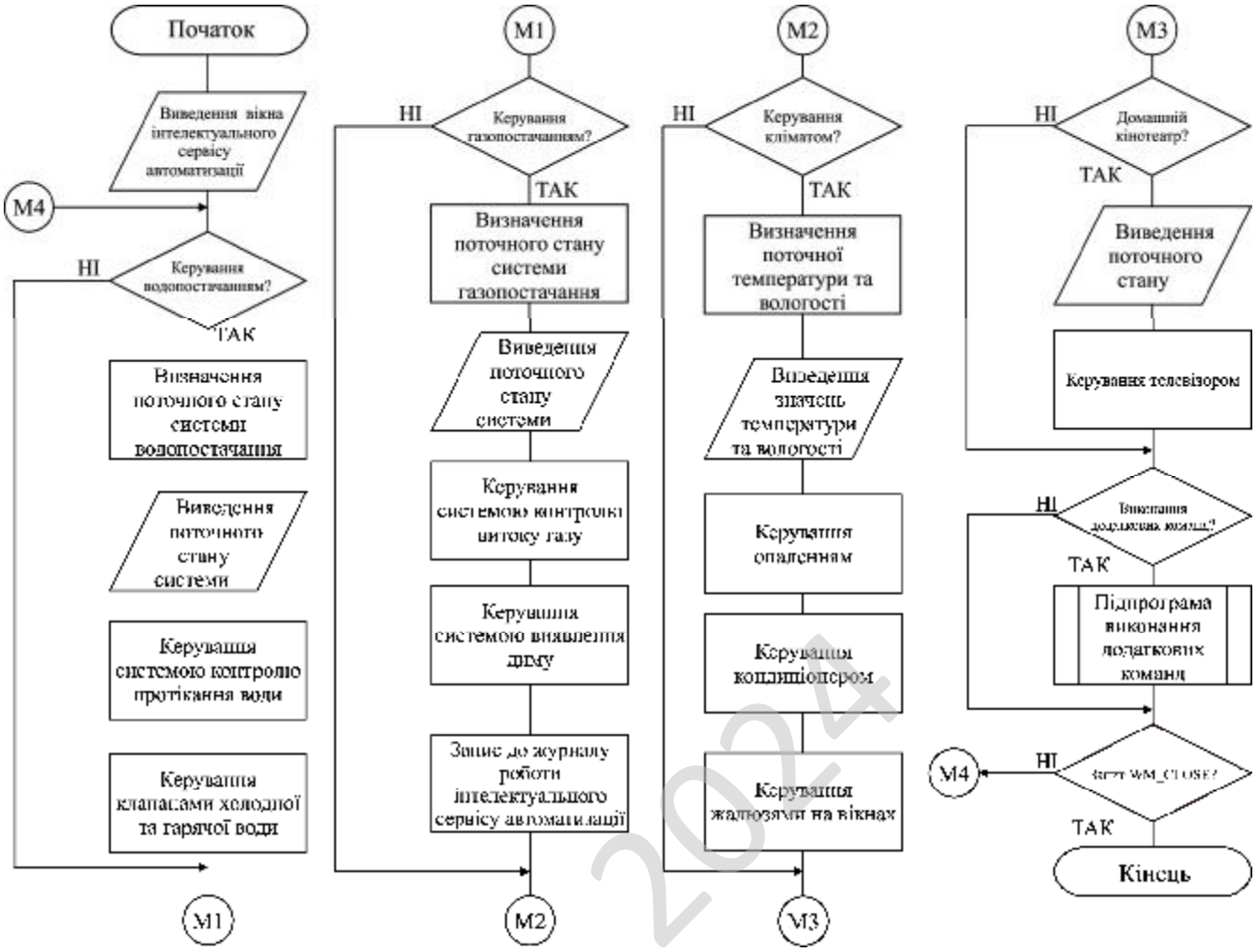

Рисунок 4.1 – Блок-схема основної програми

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції.

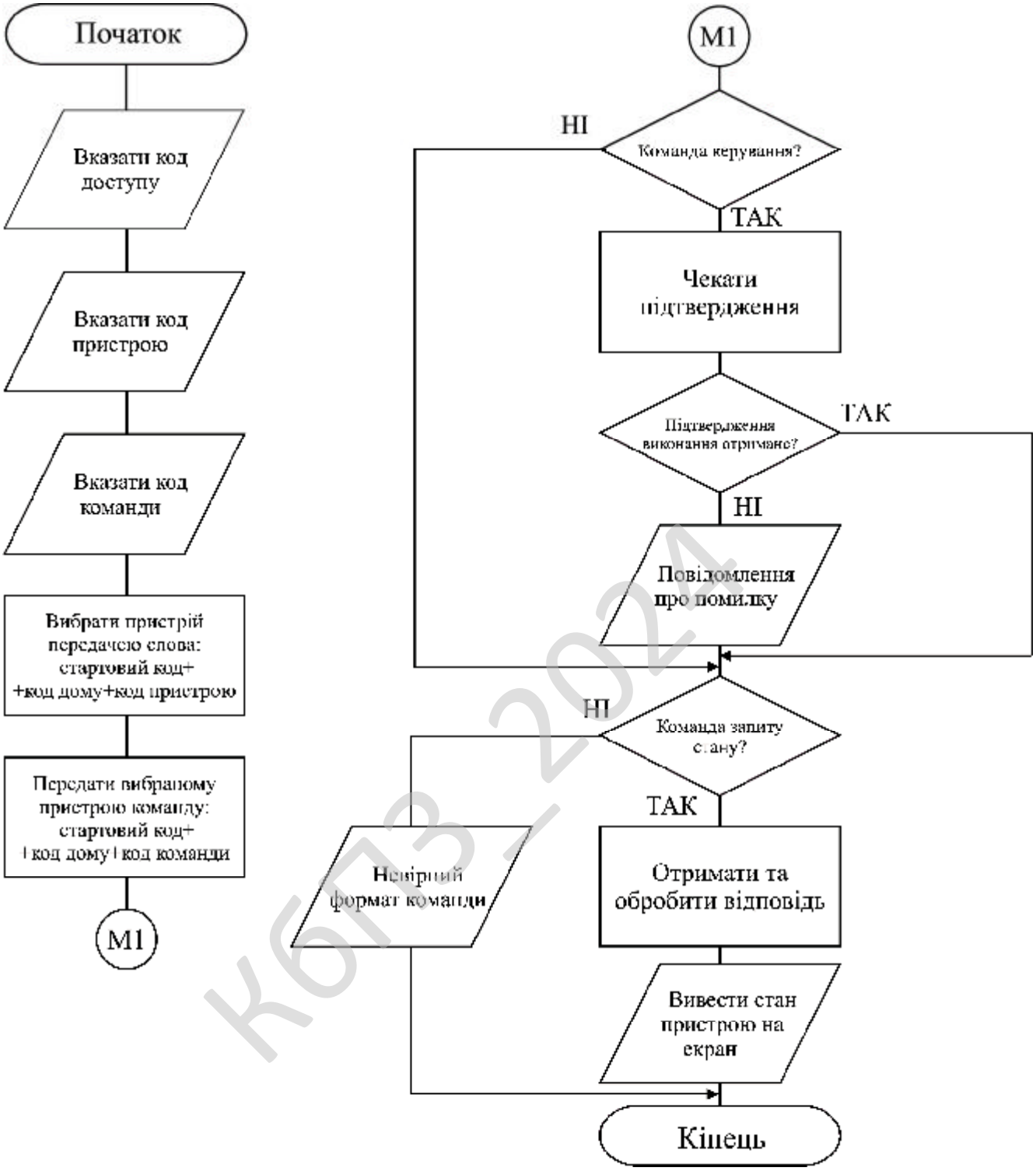

Рисунок 4.2 – Блок-схема роботи підпрограми

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями

зі стрілками, які вказують напрям впливів. Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Нижче наведемо програмний код. Який відображає підключення основних функцій у програмі.

```
//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();
}
```

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

Стандарт BAS визначає метод і протокол передачі керуючих сигналів-команд (включити, виключити, яскравіше, темніше й т.д.) по силовій електропроводці на електронні модулі, до яких підключені керовані електропобутові й освітлювальні прилади.

Усього в мережу BAS може бути об'єднане до 256 груп пристроїв з різними адресами. З погляду логіки організації внутрімережної взаємодії всі пристрої BAS можна розбити на дві більші групи: контролери й виконавчі модулі.

Контролери відповідають за генерацію команд BAS і, крім ручного

кнопкового керування, можуть мати убудований таймер або спеціалізований пристрій уведення зовнішнього впливу (датчик освітленості, фотоприймач інфрачервоного випромінювання від пульта дистанційного керування й т.д.).

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

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

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

Для передачі команди X10 потрібно одинадцять циклів (періодів) силової напруги. Перші два цикли передають стартовий код, наступні чотири цикли представляють код будинку (з A по P) і останні п'яти циклів передають код приладу (з 1 по 16) або код функції (ВМК, ВИМК і т.д.), тобто ключовий код.

Цей повний код (стартовий код + код будинку + ключовий код) завжди передається двічі безперервним блоком. Між блоками різних команд завжди повинен бути перерва в три цикли силової напруги. Виключенням із цього правила є блоки команд ЯСКРАВІШЕ/ТЕМНІШЕ, які передаються послідовно (мінімум два блоки) без затримок .

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

Нижче наведені можливі значення коду будинку й ключового коду і їхні

двійкові подання. Стартовий код – це унікальний код, завжди рівний 1110 і не має суміжних напівперіодах доповнюють, що біт в, тобто значущі біти передаються на кожний перехід силової напруги через нуль.

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

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

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

Функція Extended Code еквівалентна Extended Data: послідовність байт (без пауз), які представляють додаткові коди. Це дозволяє розроблювачам використовувати більше 256 наявних кодів.

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

Приведемо приклад.

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

```
1110010110100101011001111001011010010101100100000011100101101001011001101110010110100101100110.
```

Ця посилка містить 94 біта, і займе 47 циклів силової напруги або 0,94 с. Тому, коли ви натискаєте на кнопку ВМК, світло включається із запізнюванням. Реакція на команди "Все світло ВМК" або "Всі модулі ВИМК" помітно швидше,

тому що не передається код модуля.

Команди BAS. У протоколі BAS передбачено шість базових команд:

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

Сигнали BAS. Технологія BAS використовує цифрове подання сигналів керування. Інформація кодується двійковим кодом і передається по електричній мережі за допомогою високочастотних імпульсів.

Кожний переданий імпульс відповідає одному біту інформації сj значенням “1”. Передача чергового імпульсу відбувається в момент часу, коли сіткова напруга приймає нульове значення.

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

Більшість переданих по мережі BAS повідомлень містить, принаймні, два інформаційні поля: адреса пристрою, якому ця команда адресована, і властиво команду. Підключені до електромережі пристрої BAS приймають передані повідомлення, декодують поле адреси й, якщо він збігається з їх власним, виконують команду.

Інтеграція BAS. Основним засобом інтеграції мережі BAS із зовнішнім устаткуванням є PLC інтерфейс XM10. Будь-який контролер, що підтримує відкритий протокол обміну з XM10, може відправляти команди BAS в електромережу й, навпаки, одержувати з мережі інформацію про стан пристроїв BAS.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм RC5, який являє собою блоковий шифр із безліччю параметрів: розміром блоку, розміром ключа й числом раундів. В алгоритмі RC5 передбачені три операції: XOR, додавання й циклічні зрушення. На більшості процесорів операції циклічного зрушення виконуються за постійний час, змінні циклічні зрушення являють собою нелінійну функцію. Циклічні зрушення залежать як від ключа, так і від даних.

В RC5 використовується блок змінної довжини, але в приводиться прикладі, що буде розглянутий, 64-бітовий блок даних. Шифрування використовує $2r+2$ залежних від ключа 32-бітових слів – S_0, S_1, S_2,... S_{2r+1} – де r – число раундів. Для шифрування спочатку потрібно розділити блок відкритого тексту на два 32-бітових слова: A и B. (При впакуванні байтів у слова в алгоритмі RC5 дотримується угода про прямий порядок (little-endian) байтів: перший байт займає молодші біти регістра A й т. ін.) Потім:

$$A=A + S_0$$

$$B = B + S_0$$

Для і від 1 до r:

$$A = ((A\oplus \quad B) <<< B) + S_{2i}$$

$$B = ((B\oplus \quad A) <<< A) + S_{2i+1}$$

Вихід перебуває в регістрах A и B.

Розшифрування теж нескладно. Потрібно розбити блок відкритого тексту на два слова, A й B, а потім:

Для i від r до 1 із кроком -1:

$$B = ((B - S_{2i+1}) >>> A) \oplus A$$

$$A = ((A - S_{2i}) >>> B) \oplus B$$

$$B = B - S_i$$

$$A = A - S_0$$

Символом «>>>» позначене циклічне зрушення вправо**.** Звичайно ж**,** всі додавання й вирахування виконуються по модулю 2^{32}.

Створення масиву ключів складніше, але теж прямолінійно. Спочатку байти ключа копіюються в масив L із 32-бітових слів, доповнюючи при необхідності заключне слово нулями. Потім масив S ініціалізується за допомогою лінійного конгруентного генератора по модулю 2^{32}:

$$S_0 = P$$

Для i від 1 до $2(r + 1) – 1$:

$$S_i = (S_{i-1} + Q) \bmod 2^{32}$$

де P = 0xb7e15163 і Q = 0x9e3779b9.

Нарешті, потрібно підставити L в S:

$$i = j = 0$$

$$A = B = 0$$

Виконати $3n$ раз (де n – максимум від $2(r + 1)$ і c):

$$A = S_i = (S_i + A + B) <<< 3$$

$$B = L_i = (L_i + A + B) <<< (A + B)$$

$$i = (i + 1) \bmod 2(r +1)$$

$$i = (j +1) \bmod c$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення інтелектуальних сервісів автоматизації будинків з використанням технології BAS складається з наступних функціональних блоків:

– Блока виведення поточного стану будинку.

– Блока виведення опалення будинку.

– Блока виведення підсистеми кондиціонування.

– Блока виведення підсистем освітлення (жалюзі).

– Блока виведення температурної підсистеми.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

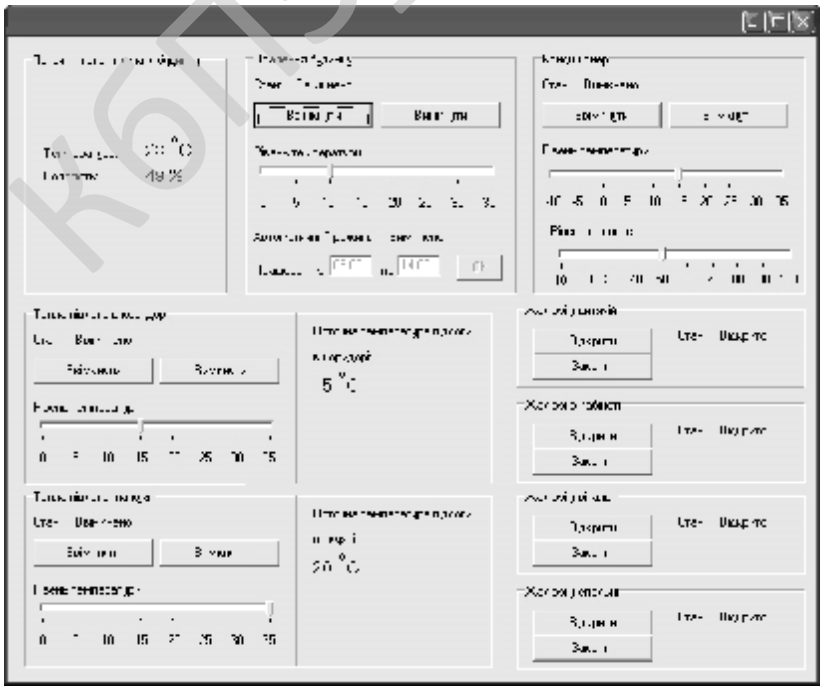

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

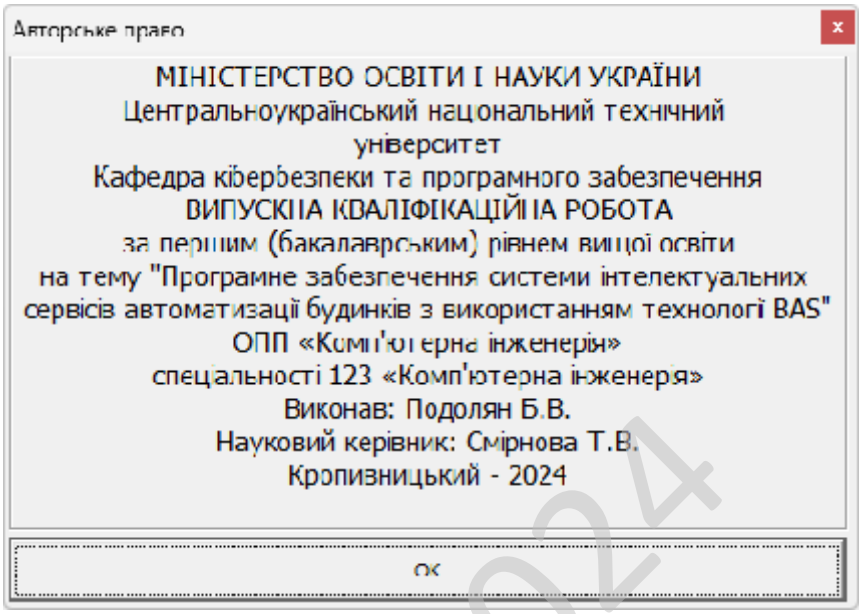

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

– Досліджена система інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

– На основі отриманих результатів досліджень створена програмна реалізація системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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 інтелектуальних сервісів автоматизації будинків з використанням технології BAS. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RC5.

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

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

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

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

5. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

6. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

7. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

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

9. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of

Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

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

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

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

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

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

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

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

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

18. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

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

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

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

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

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

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

25. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

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

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

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

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

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

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

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

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

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

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

37. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в

комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

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

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

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

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

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

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

44. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу

відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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. № 132-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS.

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 з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи інтелектуальних сервісів автоматизації будинків з використанням технології BAS;

– 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, 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 – 57 аркушів.

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.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 13.06.2024 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 інтелектуальних сервісів автоматизації будинків з використанням технології BAS

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2024 року

Основна програма

Файл Project_ihome.cpp основної програми

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

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

Файл main.cpp основної програми

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

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

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

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


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

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

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

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

//відкриття вікна "Система безпеки"
void __fastcall TForm_main::Button3Click(TObject *Sender)
```

```
{
Form_security->Show();
}
//---------------------------------------------------------------------

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

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

Файл main.h – бібліотека для файлу main.cpp

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

Файл light.cpp – керування освітленням

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

#include "light.h"
#include "CPU_BAS_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_BAS_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_BAS_(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_BAS_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_BAS_Command(0,2,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,3,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,4,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,5,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,6,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,7,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,8,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_Command(0,1,18,1);
ClientSocket->SendLeviton_BAS_(0,1,TrackBar_torch->Position);

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

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

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

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

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

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


bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
ClientSocket->Send_BAS_Command(0,8,18,1);
ClientSocket->SendLeviton_BAS_(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_BAS_Command(0,1,19,1);

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

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

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

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

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

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


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

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

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

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

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

//зміна яскравості освітлення спальні
```

```
void __fastcall TForm_light::TrackBar_bedroomChange(TObject *Sender)
{
ClientSocket->SendLeviton_BAS_(0,2,TrackBar_bedroom->Position);
}
//---------------------------------------------------------------------------

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

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

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

ClientSocket->Send_BAS_StatusQuery();

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

```
}
```

Файл light.h – бібліотека для файлу light.cpp

```
#ifndef lightH
#define lightH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm_light : public TForm
{
__published:      // IDE-managed Components
        TTrackBar *TrackBar_corridor;
        TLabel *Label11;
        TImage *Image_torch_on;
        TImage *Image_torch_off;
        TTrackBar *TrackBar_torch;
        TLabel *Label8;
        TImage *Image_corridor_on;
        TImage *Image_corridor_off;
        TTrackBar *TrackBar_drawing_room;
        TLabel *Label9;
        TImage *Image_drawing_room_on;
        TImage *Image_drawing_room_off;
        TTrackBar *TrackBar_cabinet;
        TLabel *Label12;
        TImage *Image_cabinet_on;
        TImage *Image_cabinet_off;
        TTrackBar *TrackBar_bedroom;
        TLabel *Label13;
        TImage *Image_bedroom_on;
        TImage *Image_bedroom_off;
        TTrackBar *TrackBar_baby_room;
        TLabel *Label14;
        TImage *Image_baby_room_on;
        TImage *Image_baby_room_off;
        TTrackBar *TrackBar_kitchen;
        TLabel *Label15;
        TImage *Image_kitchen_on;
        TImage *Image_kitchen_off;
        TTrackBar *TrackBar_bathroom;
        TLabel *Label16;
        TImage *Image_bathroom_on;
        TImage *Image_bathroom_off;
        TButton *Button17;
        TButton *Button18;
        TLabel *Label17;
        TLabel *torch;
        TLabel *Label19;
        TLabel *corridor;
        TLabel *Label21;
        TLabel *drawing_room;
        TLabel *Label23;
        TLabel *kitchen;
        TLabel *Label25;
        TLabel *cabinet;
        TLabel *Label27;
        TLabel *bedroom;
        TLabel *Label29;
        TLabel *baby_room;
        TLabel *Label31;
        TLabel *bathroom;
        TBevel *Bevel1;
        TLabel *Label10;
```

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

```
//---------------------------------------------------------------------------
#endif
```

Файл climate.cpp – керування кліматом

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
void __fastcall TForm_climate::TrackBar5Change(TObject *Sender)
{
ClientSocket->SendLeviton_BAS_(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_BAS_StatusQuery();
st=_BAS_State->GetStateOnOff(0, 21);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_BAS_State->GetStateOnOff(0, 22);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=_BAS_State->GetStateOnOff(0, 23);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_BAS_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=_BAS_State->GetStateOnOff(0, 26);
if(st="On") Label_5->Caption="Відкрито"; else Label_5->Caption="Закрито";
st=_BAS_State->GetStateOnOff(0, 27);
if(st="On") Label_6->Caption="Відкрито"; else Label_6->Caption="Закрито";
st=_BAS_State->GetStateOnOff(0, 28);
if(st="On") Label_7->Caption="Відкрито"; else Label_7->Caption="Закрито";
st=_BAS_State->GetStateOnOff(0, 29);
if(st="On") Label_8->Caption="Відкрито"; else Label_8->Caption="Закрито";
}
```

Файл climate.h – бібліотека для файлу climate.cpp

```
#ifndef climateH
#define climateH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------
class TForm_climate : public TForm
{
__published:      // IDE-managed Components
        TGroupBox *GroupBox2;
        TLabel *Label3;
        TLabel *Label_1;
        TButton *Button3;
        TButton *Button4;
        TTrackBar *TrackBar1;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label7;
        TGroupBox *GroupBox1;
        TLabel *Label1;
        TLabel *Label_2;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *Label16;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label27;
        TLabel *Label_tp1;
        TLabel *Label29;
        TLabel *Label30;
        TBevel *Bevel2;
        TButton *Button1;
        TButton *Button2;
        TTrackBar *TrackBar2;
        TGroupBox *GroupBox3;
        TLabel *Label31;
        TLabel *Label_3;
        TLabel *Label33;
        TLabel *Label34;
        TLabel *Label35;
        TLabel *Label36;
        TLabel *Label37;
        TLabel *Label38;
        TLabel *Label39;
        TLabel *Label40;
        TLabel *Label41;
        TLabel *Label42;
        TLabel *Label_tp2;
        TLabel *Label44;
        TLabel *Label45;
        TBevel *Bevel3;
```

```
TButton *Button5;
TButton *Button6;
TTrackBar *TrackBar3;
TGroupBox *GroupBox4;
TLabel *Label46;
TLabel *Label_4;
TLabel *Label48;
TLabel *Label49;
TButton *Button7;
TButton *Button8;
TGroupBox *GroupBox5;
TLabel *Label57;
TLabel *Label_5;
TButton *Button9;
TButton *Button10;
TGroupBox *GroupBox6;
TLabel *Label59;
TLabel *Label_6;
TButton *Button11;
TButton *Button12;
TGroupBox *GroupBox7;
TLabel *Label61;
TLabel *Label_7;
TButton *Button13;
TButton *Button14;
TGroupBox *GroupBox8;
TLabel *Label63;
TLabel *Label_8;
TButton *Button15;
TButton *Button16;
TGroupBox *GroupBox9;
TLabel *Label67;
TLabel *Label68;
TLabel *Label_t;
TLabel *Label70;
TLabel *Label71;
TLabel *Label_v;
TImage *Image8;
TImage *Image7;
TTrackBar *TrackBar4;
TLabel *Label66;
TLabel *Label73;
TLabel *Label74;
TLabel *Label75;
TLabel *Label76;
TLabel *Label77;
TLabel *Label78;
TLabel *Label79;
TLabel *Label65;
TLabel *Label23;
TTrackBar *TrackBar5;
TLabel *Label24;
TLabel *Label25;
TLabel *Label26;
TLabel *Label50;
TLabel *Label51;
TLabel *Label52;
TLabel *Label53;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label80;
TLabel *Label81;
TButton *Button17;
TLabel *Label82;
TLabel *Label83;
TEdit *Edit1;
TLabel *Label84;
TEdit *Edit2;
```

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

Файл water.cpp – керування системою водопостачання

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

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

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

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

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

//вимкнення системи контролю протікання води у ванній
void __fastcall TForm_water::Button1Click(TObject *Sender)
{
ClientSocket->Send_BAS_Command(0,9,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану холодної води у ванні
void __fastcall TForm_water::Button5Click(TObject *Sender)
{
ClientSocket->Send_BAS_Command(0,10,19,1);
```

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

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

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

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

Файл water.h – бібліотека для файлу water.cpp

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

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

Файл gas.cpp – керування системою газопостачання

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

#include "gas.h"
#include "CPU_BAS_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_BAS_Command(0,15,18,1);
}
//---------------------------------------------------------------------------

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

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

void __fastcall TForm_gas::Button3Click(TObject *Sender)
{
ClientSocket->Send_BAS_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_BAS_StatusQuery();
st=_BAS_State->GetStateOnOff(0, 15);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=_BAS_State->GetStateOnOff(0, 16);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";

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

}
```

Файл gas.h – бібліотека для файлу gas.cpp

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

Файл tv.cpp - керування дломашнім кінотеатром

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

#include <vcl.h>
#pragma hdrstop

#include "tv.h"
#include "CPU_BAS_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_BAS_(0,32,TrackBar1->Position);
}
//---------------------------------------------------------------------------


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

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

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

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

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

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

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

Файл tv.h – бібліотека для файлу tv.cpp

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

Файл phone.cpp – керування телефонним зв'язком

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

#include "phone.h"
#include "CPU_BAS_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_BAS_(0,30,TrackBar1->Position);
}
//---------------------------------------------------------------------------

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

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

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

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

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

void __fastcall TForm_phone::Button3Click(TObject *Sender)
{
ClientSocket->Send_BAS_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_BAS_StatusQuery();
```

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

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

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

Файл phone.h – бібліотека для файлу phone.cpp

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

Файл security.cpp – керування системою безпеки

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

#include "security.h"
#include "CPU_BAS_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_BAS_Command(0,17,18,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

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

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

st=_BAS_State->GetStateOnOff(0, 19);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
```

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


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

Файл security.h – бібліотека для файлу security.cpp

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

Файл 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("ДИПЛОМНИЙ ПРОЕКТ");
Memo1->Lines->Add("");
Memo1->Lines->Add("на  тему:");
Memo1->Lines->Add("");
Memo1->Lines->Add("Програмне забезпечення багатозадачної системи");
Memo1->Lines->Add("управління інтелектуальним будинком по технології X10");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("Керівник: Собінов О.Г.");
Memo1->Lines->Add("");
Memo1->Lines->Add("Розробив: студент Дяченко М.В.");
Memo1->Lines->Add("                    гр. ПМ-04");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("м. Кіровоград 2009");
}
```

Файл about.h – бібліотека для файлу about.cpp

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

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

Файл CPU_BAS_Socket.cpp – протокол X10

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

#include <basepch.h>

#pragma hdrstop

#include "CPU_BAS_Socket.h"

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

static inline void ValidCtrCheck(TCPU_BAS_Socket *)
{
    new TCPU_BAS_Socket(NULL);
}
//#include "OcelotStateUnit.cpp"
//------- -------------------
__fastcall TCPU_BAS_Socket::TCPU_BAS_Socket(TComponent* Owner)
    : TClientSocket(Owner)
{
_BAS_StateChangeNum=new T_BAS_StateChangeNum;
OcelotStateChangeNum=new TOcelotStateChangeNum;
_BAS_State=new T_BAS_State;
OcelotState=new TOcelotState;

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

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

}

__fastcall TCPU_BAS_Socket::~TCPU_BAS_Socket(void)
{
delete _BAS_StateChangeNum;
delete OcelotStateChangeNum;
delete _BAS_State;
delete OcelotState;

}
//------- -------------------
namespace CPU_BAS_Socket
{
    void __fastcall PACKAGE Register()
    {
         TComponentClass classes[1] = {__classid(TCPU_BAS_Socket)};
         RegisterComponents(" CPU-XA", classes, 0);
    }
}
//------- -------------------
void __fastcall TCPU_BAS_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_BAS_Socket::MyOnDisconnect(System::TObject* Sender,
TCustomWinSocket* Socket)
{
SendBuf("d",1);
FAuth=2;
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,FAuth);
}

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

```
TDateTime TCPU_BAS_Socket::GetTimerAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetTimer(VarNumber));
}
//------- -------------------
TDateTime TCPU_BAS_Socket::GetVarAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetVariable(VarNumber));
}
//------- -------------------
TDateTime TCPU_BAS_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_BAS_Socket::SetTimeAsVar(unsigned char VarNumber, TDateTime time)
{
  SetVariable(VarNumber,TimeToUShort(time));
}
//------- -------------------
void TCPU_BAS_Socket::SetTimeAsTimer(unsigned char VarNumber, TDateTime time)
{
  SetTimer(VarNumber,TimeToUShort(time));
}
//------- -------------------
unsigned short TCPU_BAS_Socket::TimeToUShort(TDateTime time)
{
   Word Hour, Min, Sec, MSec;
   DecodeTime(time, Hour, Min, Sec, MSec);
   return Hour*100+Min;
}
//------- -------------------
void __fastcall TCPU_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_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_BAS_Socket::GetInternalProtocolVersion(void)
```

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

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

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

void _fastcall TCPU_BAS_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_BAS_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_BAS_Socket.h - бібліотека для файлу CPU_BAS_Socket.cpp

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

#ifndef CPU_BAS_Socket
#define CPU_BAS_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_BAS_StateChangeNum {
public:           // User declarations
unsigned short Num;
void __fastcall Next(void);
__fastcall T_BAS_StateChangeNum(void);
};

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


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

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

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

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

#endif
```