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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи smart home з використанням протоколу X10”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Завірюха І.О.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

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

Завірюхі Ігорю Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи smart home з використанням протоколу X10*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи smart home з використанням протоколу X10*

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

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

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Завірюха І.О.
(прізвище та ініціали)

АНОТАЦІЯ

Завірюха І.О. Дослідження та програмна реалізація системи smart home з використанням протоколу X10. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи smart home з використанням протоколу X10.

Метою розробки є дослідження та програмна реалізація системи smart home з використанням протоколу X10.

Об'єктом дослідження є процес smart home з використанням протоколу X10.

Предметом дослідження є методи smart home з використанням протоколу X10.

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

Результат роботи – програмна реалізація системи smart home з використанням протоколу X10.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, smart home, X10

ABSTRACT

Zaviriukha I.O. Research and software implementation of the smart home system using the X10 protocol. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the smart home system using the X10 protocol.

The purpose of the development is the research and software implementation of the smart home system using the X10 protocol.

The object of the research is the smart home process using the X10 protocol.

The subject of research is smart home methods using the X10 protocol.

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

The result of the work is the software implementation of the smart home system using the X10 protocol.

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

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

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

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

Keywords: computer engineering, smart home, X10

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	–	smart home
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. У сучасному житлі вже звичайними й навіть обов'язковими стають різні пристрої й пристосування, що забезпечують високу якість життя й комфорт мешканців. І якщо раніше канальні кондиціонери, електроуправляємі водопровідні вентилі, індивідуальне управління кліматом і безліч світлових груп були атрибутами великого заміського будинку, то зараз ці особливості властиві й міській квартирі. Зрозуміло, при повсякденній експлуатації настільки різноманітного встаткування в користувачів виникає безліч повторюваних завдань, наприклад:

– одночасне вимикання множини світлових груп;

– вимикання великої кількості приладів при відході із квартири й включення їх у потрібний режим при поверненні у квартиру;

– управління кліматом у приміщеннях залежно від часу доби й присутності людей;

– контроль цілісності комунікацій і справності встаткування.

У першу чергу, саме для виконання таких рутинних операцій, причому з великим ступенем надійності й точності, призначені системи домашньої автоматики, тобто системи «Smart home».

Але логічно, що саме все не буде включатися й відключатися у smart home (інтелектуальному домі). Для того, щоб smart home запрацював, по-перше необхідно установити відповідне устаткування, по-друге необхідно з’єднати це устаткування у мережу, й відповідно написати якесь програмне забезпечення, яке повинно керувати устаткуванням.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи smart home з використанням протоколу X10.

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

– Огляд існуючих систем smart home з використанням протоколу X10.

– Дослідження системи smart home з використанням протоколу X10.

– Програмна реалізація системи smart home з використанням протоколу X10.

Об'єктом дослідження є процес smart home з використанням протоколу X10.

Предметом дослідження є методи smart home з використанням протоколу X10.

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

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

– Удосконалено метод smart home з використанням протоколу X10.

– Розроблено вітчизняний продукт smart home з використанням протоколу X10, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі smart home з використанням протоколу X10.

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

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи smart home з використанням протоколу X10, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Smart home – Інтелектуальним можна назвати будинок, оснащений засобами автоматичного контролю над всіма системами життєзабезпечення.

На певні впливи усередині будинку й/або за його межами ми одержуємо відгук відповідно до заданого алгоритму. Одержувана послідовність відгуків нагадує розвиток подій по певному сценарії, що написаний спеціально для конкретного об'єкта з урахуванням індивідуальних побажань замовника. Тобто мова йде про управління ІБ за допомогою автоматизованих комплексів, про автоматизацію будинку, термін для багатьох більше зрозумілий і звичний.

Одним з основних компонентів інтелектуального будинку є система автоматизованого управління експлуатацією будинку (АСУ). АСУ експлуатації будинку – це комплекс програмно-апаратних засобів, основною задачею якого є забезпечення надійного й гарантованого управління всіма системами, що перебувають в експлуатації будинку, і виконавчими пристроями. Система здатна за рахунок повної нероз'єднаної інформації від всіх експлуатованих підсистем прийняти правильне рішення й виконати відповідну дію, проінформувати відповідну службу про подію.

Поряд із широким використанням Інтернету й мобільного зв'язку, найпоширенішим засобом управління є Оперативна диспетчерська система (ОДС) житлового комплексу. ОДС – сполучна ланка при інтеграції підсистем всілякого призначення через зручність спостереження за станом інженерних підсистем і за безпекою ІБ у цілому. ОДС має на увазі інтеграцію систем всілякого профілю й призначення.

Оперативно-диспетчерська система (ОДС) призначена й створюється для моніторингу в безперервному режимі й управління інженерними системами,

системами забезпечення безпеки й системами обліку, для запобігання аварійних ситуацій, відстеження небезпечних ситуацій з наступним оповіщенням, реєстрації подій, ведення журналу й дистанційного управління виконавчими механізмами.

При створенні ОДС житлового комплексу як стандарт при автоматизації будинків найбільше широко застосовується технологія LonWorks. Вона забезпечує реальний виграш у плані зниження вартості монтажу, пуско-налагодження й підвищення темпів монтажних робіт. Надається можливість поетапної реалізації й розширення функцій без проведення дорогих електромонтажних робіт, при цьому працездатність уже встановленої частини системи при цьому не порушується.

Розподілена система моніторингу параметрів функціонування й управління всіма технічними системами, а також оповіщення про позаштатні ситуації здійснюється за допомогою програмно-математичного забезпечення (ПМО) диспетчера ОДС.

Оперативна диспетчерська система має ієрархічну багаторівневу структуру. На границі кожного із шарів існує деякий інтерфейс взаємодії пристроїв вищого й нижчого рівнів. При цьому в кожної підсистеми ці інтерфейси в загальному випадку власні й переважає підпорядкування пристроїв нижнього рівня пристроям верхнього, тобто "знизу" надходить інформація, а "нагорі" приймають рішення.

Рівень 1 – Система кабельних комунікацій (СКК)**.** Призначається для організації мереж і забезпечує універсальність і гнучкість проектних рішень, зручність адміністрування й розширюваність системи в майбутньому.

Рівень 2 – Рівень периферійних пристроїв: первинні датчики й виконавчі пристрої (зчитувачі, заслінки, нагрівачі, звіщувачі, відеокамери й т.п.), а також пристрої узгодження сигналів первинних датчиків із входами контролерів збору інформації. Датчики на встаткуванні підсистем замірять параметри вузлів і при відхиленні від припустимих значень передають сигнал на контролери.

Рівень 3 – рівень керуючого устаткування: контролери збору інформації (віддалені модулі вводу-виводу), програмувальні логічні контролери (ПЛК), інтелектуальні панелі управління встаткуванням, робочі станції управління інженерними системами. Контролери, розташовані в шафах автоматики, управляють роботою периферійного встаткування, збирають і передають на АРМ (автоматизоване робоче місце) диспетчера всю інформацію. ПМО концентраторів повинне забезпечувати відновлення мережних змінних не рідше 1 рази в секунду.

Рівень 4 – рівень інтеграції підсистем: центральний сервер ОДС і АРМ.

Сервер ОДС містить засоби організації обміну інформацією з контролерами збору інформації (по керуючих шинах), а так само спеціалізоване програмне забезпечення на базі SCADA-системи для збору й архівування інформації, що надходить від інженерних систем.

АРМ (автоматизоване робоче місце) диспетчера (комп'ютер або комп'ютерна мережа зі спеціалізованим програмним забезпеченням, що дозволяє систематизувати прихожу інформацію від контролерів, видавати її в зручній для сприйняття формі, управляти периферійними пристроями й програмувати відгук системи на певну послідовність подій).

Інформація про стан всіх підсистем об'єкта відображається на екрані АРМа за допомогою екранних форм (ЕФ) у зручному й наочному для диспетчера ОДС виді.

Можливість управління всіма системами об'єкта з однієї робочої станції істотно знижує витрати на навчання персоналу ОДС.

1.2 Область застосування

Система "smart home" є комбінацією різних систем слабких струмів і силової електропроводки. Однією з яскравих функцій системи smart home є управлінням освітленням, основа цінності цієї функції є програмувальні вимикачі, тобто ви можете запрограмувати управління будь-якого світла або груп

світла за допомогою будь-якого вимикача, а також створювати сценарії, управляти яскравістю світла, імітувати присутність шляхом автоматичного включення світла під час вашої відсутності.

Другим по полярності об'єктом управління smart home є створення систем клімат-контролю, це мабуть сама складна частина, тому що тут мова йде про цілий комплекс пристроїв, таких як: батареї опалення, кондиціонери, теплі підлоги, широкі стекла через які йде інтенсивне нагрівання в теплий час доби. Все це треба погодити в єдину логіку роботи для створення справжній клімат контролю зі стійкою заздалегідь заданою температурою.

Так само варто розглянути систему життєзабезпечення будинку, вона представлена в розумному будинку у вигляді систем витоку води, пожежною системою, датчиком падіння критичної температури, датчиком освітленості. Все це дозволяє залишати ваш будинок у цілості й схоронності під час вашої відсутності. Система безпеки напевно містить у собі найбільший список засобів і встаткування по можливості інтеграції в smart home. Список починається з відеоспостереження й закінчується системами доступу за біометричним даними людини.

Система управління smart home припускає наявність пультів управління, планшетних комп'ютерів, смартфонів й інших пристроїв.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи smart home з використанням протоколу X10, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

У рамках огляду технологій побудови інтелектуальних будинків розглянемо такі технології як KNX/EIB і X10/Adicon.

Технологія KNX/EIB

Як приклад можна розглянути проект побудови автоматизованої системи управління й безпеки «Smart home» (АСУБ ІБ). На об'єкті створювалися наступні системи:

- електропостачання, освітлення й управління приводами;
- управління інженерним устаткуванням;
- управління опаленням, вентиляцією й кондиціюванням;
- охоронно-пожежної сигналізації;
- домашньої кабельної мережі;
- відеодомофоного зв'язку й відеоспостереження;
- звукової трансляції й оповіщення;
- місцевого, дистанційного й централізованого управління.

АСУБ ІБ створювалася для впровадження єдиної інтелектуальної взаємозалежної системи управління різними електроспоживачами, інженерними пристроями, мультимедіа-апаратурою й підсистемами безпеки житлової квартири для забезпечення:

- зручного управління освітленням;
- зручного управління побутовий аудио-, відеотехнікою;
- зручного використання різних інженерних пристроїв;
- комфортного й безпечного проживання у квартирі;

– економії електроенергії, води.

Для зв'язку частини виконавчих і керуючих пристроїв між собою застосована європейська інсталяційна шина (EIB). Застосування такого інтерфейсу дозволяє використовувати дуже широкий спектр керуючих приладів (клавішних панелей) різноманітного дизайну. На стадії впровадження АСУБ ІБ у компоненти системи на шині EIB записуються мікропрограми, що забезпечують виконання даною підсистемою задач, передбачених ТЗ. Крім того, для об'єднання встаткування АСУБ ІБ використовуються інші стандартні інтерфейси – RS-232, RS-485, LonWorks.

Ключовим компонентом створеної системи є центральний сервер, призначений для:

– об'єднання різноманітного встаткування АСУБ ІБ у єдину інформаційну мережу;

– підтримки графічного користувальницького інтерфейсу;

Центральний сервер працює під управлінням програмного забезпечення «Комфортний Будинок», настроєного для рішення задач автоматизації й управління відповідно до ТЗ. Використання центрального сервера дозволяє знизити витрати на встаткування шляхом вибору компонентів з необхідними характеристиками й мінімальною вартістю, з наступним об'єднанням їх у логічну мережу. Це можливо завдяки підтримці ПЗ «Комфортний Будинок» протоколів обміну для широкого спектра встаткування.

Основа підсистеми управління освітленням і приводами – шинні компоненти EIB. Реле для шини EIB призначені для комутації світлових груп і управління приводами штор і екрана домашнього кінотеатру. Для регулювання яскравості світлових груп застосовуються потужні багатоканальні диммеры, керовані центральним сервером. В одній з кімнат змонтована «світлова стеля» з люмінесцентними лампами, які диммуються спеціальними електронними баластами. Баласти управляються комбінованим пристроєм на шині EIB.

До шини EIB підключені також:

– настінні клавішні панелі, призначені для місцевого управління, у тому числі управління освітленням і приводами штор, екрана;

– датчики руху, призначені для визначення присутності людини у відповідній зоні й подачі команд виконавчим пристроям.

Команди із клавішних панелей і датчиків руху надходять до реле безпосередньо по шині EIB, а до диммерів – через центральний сервер. Крім того, у системі використовуються автономні датчики присутності – для управління світловими групами в підсобних приміщеннях. Ці датчики безпосередньо (вбудованим реле) включають необхідні світлові групи при виявленні руху людини. За допомогою спеціального датчика система при включенні проектора опускає екран домашнього кінотеатру й плавно гасить світло в кінозалі. При вимиканні проектора плавно включається світло й через якийсь час піднімається екран.

Основні функції підсистеми управління інженерним устаткуванням:

– перемикання режимів водопостачання;

– визначення й запобігання аварійного витоку води;

– комутація груп розеток загального призначення;

– комутація нагрівальних елементів «тепла підлога».

У розглянутій квартирі є кілька санвузлів, згрупованих навколо трьох сантехнічних «стояків». У кожній такій зоні встановлені електричні накопичувальні водонагрівачі й три вентилі з електроприводом. Також передбачений датчик температури магістральної гарячої води. При відключенні центрального гарячого водопостачання система переводить вентилі в режим подачі гарячої води від електронагрівника й включає нагрівач у прискорений режим нагрівання. При поновленні нормального центрального постачання гарячою водою встаткування переводиться у звичайний режим.

Коли квартиру залишають всі мешканці, серед інших дій відключаються водонагрівачі й закриваються всі вентилі, перекриваючи подачу води у квартиру.

Це робиться для запобігання аварійного вибігу у відсутності людей у квартирі. Установлені в санвузлах і на кухні датчики вибігу визначають наявність води на підлозі приміщення. У результаті керованими вентилями перекривається подача води в зоні, де виявлен вибіг. При цьому формується тривожний сигнал і повідомлення на центральній панелі. Дана підсистема також відключає більшість електричних розеток і «теплі підлоги» при тривалій відсутності людей у квартирі. Підсистема управління опаленням, вентиляцією й кондиціюванням (ОВК) призначена для:

– управління роботою системи приточної вентиляції;

– управління роботою системи кондиціювання;

– управління приводам клапанів радіаторів опалення з метою підтримки заданих кліматичних умов і режимів роботи кліматичного встаткування.

Бажана температура в приміщеннях установлюється користувачем при виборі режиму роботи підсистеми ОВК відповідно до вимог ТЗ. Установлене значення надходить у відповідний внутрішній блок системи кондиціювання й у регулятор потужності калорифера системи вентиляції. Таким чином, з урахуванням обраних режимів у різних приміщеннях система вентиляції забезпечує підігрів зовнішнього повітря до мінімального значення, передбаченого активними режимами, а доведення температури повітря в приміщеннях до потрібного значення здійснюються за допомогою радіаторів опалення й системи кондиціювання.

Режими роботи підсистеми ОВК відповідно до ТЗ перемикаються автоматично центральним сервером по складених часових розкладах або при виборі сценаріїв. При цьому передбачена можливість блокування обраного режиму, що забороняє автоматичну зміну режиму до розблокування. Користувачі в будь-який час можуть змінити температуру в приміщеннях, швидкості вентиляторів кондиціонерів і швидкість вентилятора приточної системи. Такі налаштування здійснюються із центральної панелі управління. При

бажанні нові налаштування можуть бути записані як постійні для поточного режиму.

Домашня підсистема охоронно-пожежної сигналізації (ОПС) створюється з метою:

– оповіщення про пожаронебезпечні ситуації й несанкціоноване проникнення.

– можливість підключення ОПС на пульт спеціальних служб швидкого реагування (далі Спецоператор) для тривожного оповіщення й безперервного моніторингу працездатності системи.

Охоронно-пожежна сигналізація об'єкта спроектована на базі приймально-контрольного приладу «Віста-101». Основний блок підсистеми й всі модулі розширення розташовані в технічній зоні коридору. Електроживлення комплексу здійснюється через блок захисту від перенапруг. Резервне живлення здійснюється від акумуляторної батареї; при цьому забезпечується працездатність комплексу протягом доби в черговому режимі й не менш трьох годин у режимі тривоги.

Охоронні повідомлювачі розташовуються у всіх приміщеннях об'єкта, що мають вікна й зовнішні двері. В ОПС в основному використовуються охоронні комбіновані ІЧ-СВЧ повідомлювачі, що дозволяє запобігти помилковим спрацьовуванням повідомлювачей при наявності теплових потоків від систем вентиляції й кондиціювання. У якості пожежних повідомлювачей обрані димові й теплові граничні повідомлювачі. Димові повідомлювачі встановлені в місцях найбільш імовірного поширення диму (у коридорах поблизу каналів витяжної вентиляції). Тепловий повідомлювач установлений на кухні. Вибір цього типу повідомлювача визначається тим, що в нормальному режимі для кухні можлива поява деякого задимлення; у теж час при загорянні найбільш імовірне усього різке підвищення температури. Стан пожежних повідомлювачей контролюється системою постійно й не залежить від режиму роботи охоронної сигналізація.

Для інформування людей про пожежу й тривожні ситуації служать звукові оповіщувачі (сирени). Управління підсистемою ОПС здійснюється за допомогою

виносного пульта, розташованого поблизу вхідних дверей. Крім того, можливе управління підсистемою ОПС із бездротових пультів і брелоків. Підсистема ОПС взаємодіє із іншим устаткуванням, передаючи в АСУБ ІБ сигнали про постановку на охорону, зняття з охорони й тривоги. При постановці в режим повної охорони (коли у квартирі не можуть перебувати люди), АСУБ ІБ виключає все світло, відключає розетки, теплі підлоги, кліматичне встаткування перекриває подачу води – тобто виконує всі дії, необхідні для забезпечення максимальної безпеки житла, залишеного без догляду.

Після реєстрації тривоги в будь-якому приміщенні квартири приймально-контрольний прилад по заданому для конкретної події алгоритму включає сигнали оповіщення й передає на пульт служби реагування повідомлення про подію, що відбулася. Крім того, на пульт служби реагування передаються повідомлення про несправності в підсистемі ОПС.

Підсистема домашньої кабельної мережі (ДКС) розроблена на базі технології LexCom Home групи компаній Lexel. Основною перевагою побудованої мережі є її універсальність – ті самі комунікаційні розетки в процесі експлуатації можуть бути призначені як телефонні, комп'ютерні або телевізійні. Таке налаштування може виконати користувач системи простою перестановкою комутаційних кабелів у шафі автоматики. одатковою функцією ДКС є можливість трансляції аудіо-відеосигналів від побутових відтворюючих апаратів у телевізійну мережу квартири з підтримкою ІЧ-дистанційного управління. Таким чином, користувачі можуть користуватися, наприклад, супутниковим приймачем або DVD-плеєром, які знаходяться в одній з кімнат, на телевізорах в інших кімнатах, так, як якби ці апарати перебували безпосередньо біля телевізора.

Устаткування підсистеми відеодомофоного зв'язку й відеоспостереження забезпечує виконання наступних основних функцій:

– двунаправлений аудіозв'язок із відвідувачами квартири;

– відеоспостереження за відвідувачами квартири й за обстановкою перед вхідними дверима;

– акустичний контроль (прослуховування) обстановки перед вхідними дверима у квартиру;

– архівування відеоінформації, відтворення раніше записаної відеоінформації;

– трансляція зображення від викличної панелі в телевізійну мережу квартири.

Виклична панель надає можливість відвідувачеві викликати мешканців квартири й переговорюватися з ними з використанням вбудованих у панель гучномовця й мікрофона. Крім того, у викличну панель вбудована кольорова камера, зображення з якої надходить на абонентський блок (монітор) домофону й на вхід центрального сервера для архівації. На центральній панелі управління при надходженні виклику від відвідувача квартири автоматично на певний час з'являється зображення від камери викличної панелі. Відповідь на виклик відвідувача здійснюється спеціальною кнопкою на моніторі домофону. Крім того, при відсутності виклику натискання цієї кнопки приводить до висновку зображення й звуку від викличної панелі відповідно на монітор і гучномовець абонентського блоку (режим контролю обстановки).

На площадці перед вхідними дверима квартири приховано встановлена відеокамера, призначена для візуального контролю обстановки. Сигнал від камери надходить на вхід відеопідсистеми центрального сервера для архівування. Зображення від цієї камери, як і зображення від камери викличної панелі, може бути виведене на центральну панель управління в будь-який час. Передбачено спеціальний адаптер, що забезпечує можливість переговорів мешканців з відвідувачем через звичайні телефони, підключені до телефонної мережі квартири. Також адаптер дублює сигнал виклику від викличної панелі в телефонну мережу квартири. Крім того, спеціальний відеомодулятор перетворить відеосигнал від камери викличної панелі в телевізійний сигнал і подає його в телевізійну мережу квартири. Таким чином, мешканці квартири можуть

переговорюватися з відвідувачами й спостерігати за ними, не наближаючись до монітора видеодомофону.

Відеопідсистема центрального сервера складається із плати відеозахоплення в центральному сервері й спеціалізованому програмному забезпеченні. Робота з відеоінформацією (перегляд, пошук, архівування) здійснюється користувачем у середовищі єдиного графічного користувальницького інтерфейсу системи. У квартирі реалізована підсистема звукової трансляції й оповіщення (ЗТО), призначена для:

– відтворення сигналів від центральних джерел звукових сигналів у різних приміщеннях квартири;

– трансляції голосових повідомлень із одного приміщення квартири в інші.

Джерела звукового сигналу – медіа-сервер і тюнер – підключені до багатоканального підсилювача-комутатора. До виходів підсилювача-комутатора підключаються акустичні системи, що вбудовуються, у зонах трансляції.

Обрані джерела забезпечують можливість прослуховування одночасно до п'яти незалежних фонограм – дві радіостанції за допомогою тюнера й три фонограми за допомогою медіа-сервера. Управління підсистемою ЗТО здійснюється:

– зі спеціалізованого пульта, установленого в одній з кімнат;

– с сенсорних клавішних панелей, розміщених у зонах трансляції;

– с центральної панелі управління.

В одній із зон трансляції (Кінозалі) управління встаткуванням ЗТО не передбачено. Акустична система в цій зоні призначена лише для відтворення голосових повідомлень.

Трансляція голосових повідомлень здійснюється за допомогою телефонних апаратів, підключених до ДКС. При цьому користувач відповідними командами тонового режиму переводить АТС у режим «Пейджинг» і проговорює в мікрофон телефонного апарата повідомлення. Звуковий сигнал при

цьому відтворюється через гучномовці всіх зон. Основні задачі, розв'язувані підсистемою місцевого, дистанційного й центрального управління:

– прийом і обробка користувальницьких команд управління освітленням, приводами, звуковою трансляцією, кліматичними режимами з :

1) настінних клавішних панелей,
2) пультів дистанційного управління,
3) центральної панелі управління;

– організація інформаційного обміну між іншими підсистемами;

– надання користувачеві зручного і єдиного для різних підсистем графічного інтерфейсу.

У якості центрального керуючого компонента обрана система «Комфортний будинок» компанії СРС. Дана система має модульну структуру й дозволяє інтегрувати широкий спектр устаткування. До складу підсистеми управління входять:

– центральний сервер (відеосервер);
– центральна панель управління – сенсорний РК-монітор;
– ІЧ-приймачі;
– настінні клавішні панелі;
– інше (допоміжне) устаткування;
– набір необхідних програмних компонентів.

Центральний сервер здійснює необхідну координацію всіх підсистем АСУ, забезпечує функціонування відеопідсистеми. Як основний пристрій управління й візуалізації – центральної панелі управління – обраний рідкокристалічний монітор настінного кріплення із сенсорним екраном. ІЧ-приймачі одержують інфрачервоні сигнали від пультів дистанційного управління й перетворять їх у повідомлення, передані центральному серверу у встановленому форматі по інтерфейсі RS-485.

Для живлення центрального сервера, центральної панелі управління й ряду інших пристроїв призначене джерело безперебійного живлення (ДБЖ). При

аварійних відключеннях обраний ДБЖ може самостійно коректно виключити центральний сервер і включити його з появою живлячої напруги. Настінні клавішні панелі призначені для місцевого (локального) управління користувачем устаткуванням АСУБ ІБ. Клавішні панелі з'єднані інформаційним кабелем – шиною EIB між собою й з іншими EIB-компонентами. Функції клавішних панелей призначаються при програмуванні системи відповідно до ТЗ.

Програмне забезпечення, установлене на центральному сервері, вирішує наступні основні задачі:

– моніторинг стану й управління встаткуванням різних підсистем АСУБ ІБ;

– передача керуючих команд між різним устаткуванням;

– візуалізація й управління АСУБ ІБ у цілому;

– відображення, запис відеоінформації, відтворення раніше записаної відеоінформації від камер підсистеми відеоспостереження;

– групове й автоматичне управління встаткуванням АСУБ ІБ (виконання макросів і сценаріїв);

– обробка й виконання команд, прийнятих від ІЧ-приймачів.

Центральний сервер може бути підключений до комутатора підсистеми ДКС. При цьому доступ до центрального сервера (і, відповідно, можливість візуалізації й управління АСУБ ІБ) може бути отриманий удаленно через мережу Інтернет зі стандартного Web-браузера.

ІЧ-приймачі розташовуються по одному в приміщенні, де потрібне дистанційне управління й підключаються в спеціальні настінні розетки. ІЧ-приймачі поєднуються в єдину мережу з інтерфейсом RS-485 і підключаються до центрального сервера. Для роботи з ІЧ-приймачами придатні будь-які ІЧ пульти дистанційного управління (ПДУ), що працюють по протоколах Sony або Sharp. При виборі пультів перевага варто віддати тим універсальним ПДУ, яких навчають.

На рисунку 2.1 зображені елементи інтелектуального будинку.

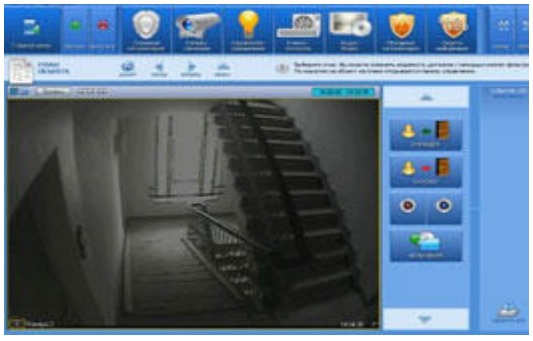
а) підсистема управління освітленням і приводами

б) підсистема охоронно-пожежної сигналізації

в) підсистема управління опаленням, вентиляцією й кондиціюванням

г) підсистема домашньої кабельної мережі

д) програмне забезпечення, установлене на центральному сервері



е) підсистема відеодомофоного зв'язку й відеоспостереження

Рисунок 2.1 – Елементи системи управління інтелектуальним будинком на основі технології KNX/EIB

На етапі впровадження АСУБ ІБ для рішення задач, передбачених ТЗ, здійснювалася:

– розробка користувальницького інтерфейсу,

– установка й конфігурування відеопідсистеми,

– конфігурування програмних модулів для міжсистемної взаємодії,

– налаштування сценаріїв і макросів;

– конфігурування підсистеми дистанційного управління.

Технологія X10

Проект являє собою автоматизовану систему управління й безпеки Smart home, розроблену Лабораторією I-Home.ru по індивідуальному замовленню для оснащення житлової квартири. Для рішення задач управління як платформа автоматизованої системи електротехнічним устаткуванням застосована технологія X10. Ця технологія використовує для передачі даних побутову електричну проводку. Її відмінність від інших технологій полягає в тім, що для наступної модернізації не потрібен великий об'єм монтажних робіт.

Основним функціональним вузлом інтелектуальної мережі будинку є універсальний побутовий контролер "OCELOT". У рамках поставлених задач він виконує наступні функції:

– контроль за вибігом води (моніторинг датчиків вибігу і перекриття електромеханічних відсічних клапанів води);

– контроль за вологістю в сантехнічних приміщеннях (моніторинг датчиків вологості й забезпечення необхідного режиму роботи вентиляторів витяжки);

– контроль за системою витяжки на кухні (моніторинг роботи кухонної плити за допомогою датчиків струму);

– відключення теплих підлог, якщо за останню добу була відсутня живаючи активність у приміщеннях (моніторинг датчиків руху, дати й часу);

– контроль за сигналами X10 у силовій проводці й виконання відповідних макросів (дистанційне ручне управління кухонною витяжкою, світлові схеми, управління теплою підлогою й т.п.).

Охоронні функції реалізовані на основі бездротової охоронно-пожежної сигналізації PowerMax. Особливістю даної системи є можливість управління пристроями X10 як дистанційно (з телефону) так і локально (із клавіатури, по датчику руху й т.п.). Це дозволяє дистанційно включати, виключати прилади й освітлення, імітувати присутність хазяїв (включаючи освітлення по таймеру), автоматично увімкнути світло в прихожій по датчику руху й т.п.

Даний проект дозволяє реалізувати наступну функціональність:

– незалежне управління кожною групою освітлення із клавішних панелей;

– плавне регулювання яскравості світлових груп;

– управління освітленням автоматично по сигналах датчиків руху;

– управління освітленням автоматично по заздалегідь складеній програмі;

– сценарне управління освітленням;

– дистанційне (з ІЧ-пультів) управління освітленням, електроприладами, побутовою технікою;

– автоматичне й ручне управління кухонною витяжкою;

– управління розеточними групами, нагрівальними елементами «тепла підлога» (у режимі «включене/виключено»);

– визначення вибігу води в санвузлах і на кухні;

– управління відсічними водопровідними клапанами:

1) по сигналах від датчиків вибігу,

2) командам користувача,

3) заздалегідь складеній програмі;

– охорону квартири від вторгнень і загорянь із оповіщенням користувачів про тривоги по телефоні.

а) загальний вид системи

б) універсальний побутовий контролер "OCELOT"

в) бездротова охоронно-пожежна сигналізації PowerMax

Рисунок 2.2 – Елементи системи управління інтелектуальним будинком на основі технології X10/Adicon

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++

досить просте в користуванні, його вихідний код значно менше по об’єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

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

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

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

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими

функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

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

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи smart home з використанням протоколу X10.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Огляд технології X10

Технологія домашньої автоматизації X10 призначена для інтелектуального управління освітленням і побутовими електроприладами з використанням силової електропроводки.

Сигнали управління X10 передаються й приймаються безпосередньо по електричній мережі (Power Line Carrier або PLC). Відсутність додаткових дротів робить інсталяцію систем X10 швидкою й легкою.

Для передачі команд управління в системах X10, крім силової проводки, використовується радіоканал із частотою 433Мгц. Так забезпечується комфорт бездротового способу управління.

Одна з головних переваг технології X10 у порівнянні із традиційною електроінсталяцією полягає в тому, що схема управління може багаторазово мінятися простим переналаштуванням окремих компонентів мережі X10 – контролерів, вимикачів, реле й діммерів. У випадку традиційної електроінсталяції зв'язок «орган управління – об'єкт управління» задаються жорстко раз і назавжди структурою кабельної проводки. Системи X10 легко розширювані. У будь-який момент у діючу інсталяцію можна додати нові компоненти навіть якщо опоряджувальні роботи давно закінчені.

Вартість мінімального комплекту встаткування X10 становить 1-2 тис. грн., що робить технологію доступною й популярною.

Технологія X10 веде свою історію з початку 70-х років минулого століття. У той час інженери з англійської компанії PICO Electronics уперше створили пристрої, що використовували електричну проводку як середовище мережної

комунікації. З тих пор технологія безупинно вдосконалювалася й сьогодні автоматикою X10 оснащені мільйони будинків по усьому світі. Особливе поширення й популярність технологія одержала в країнах Північної Америки.

Управління X10

За допомогою технології X10 вирішуються завдання комфортного управління освітленням, компонентами інженерних систем і побутових електроприладів. Технологія X10 забезпечує різноманітні способи управління домашнім устаткуванням у ручному й автоматичному режимах. Додатковий асортименти мультимедійних і охоронних пристроїв розширює функціональні можливості базового комплекту X10 по управлінню освітленням і електроприладами й робить всю лінійку ще більш привабливої для самого широкого кола споживачів.

Рисунок 3.1 – Схема управління технології X10

Функції X10

Устаткування X10

Мережа X10 – це сукупність контролерів і виконавчих пристроїв, взаємодіючих один з одним по електричній мережі або радіоканалу. При необхідності додатково використовуються спеціальні системні пристрої, що забезпечують працездатність мережі X10 у цілому. Устаткування X10 може задовольнити найрізноманітнішим вимогам до розміщення.

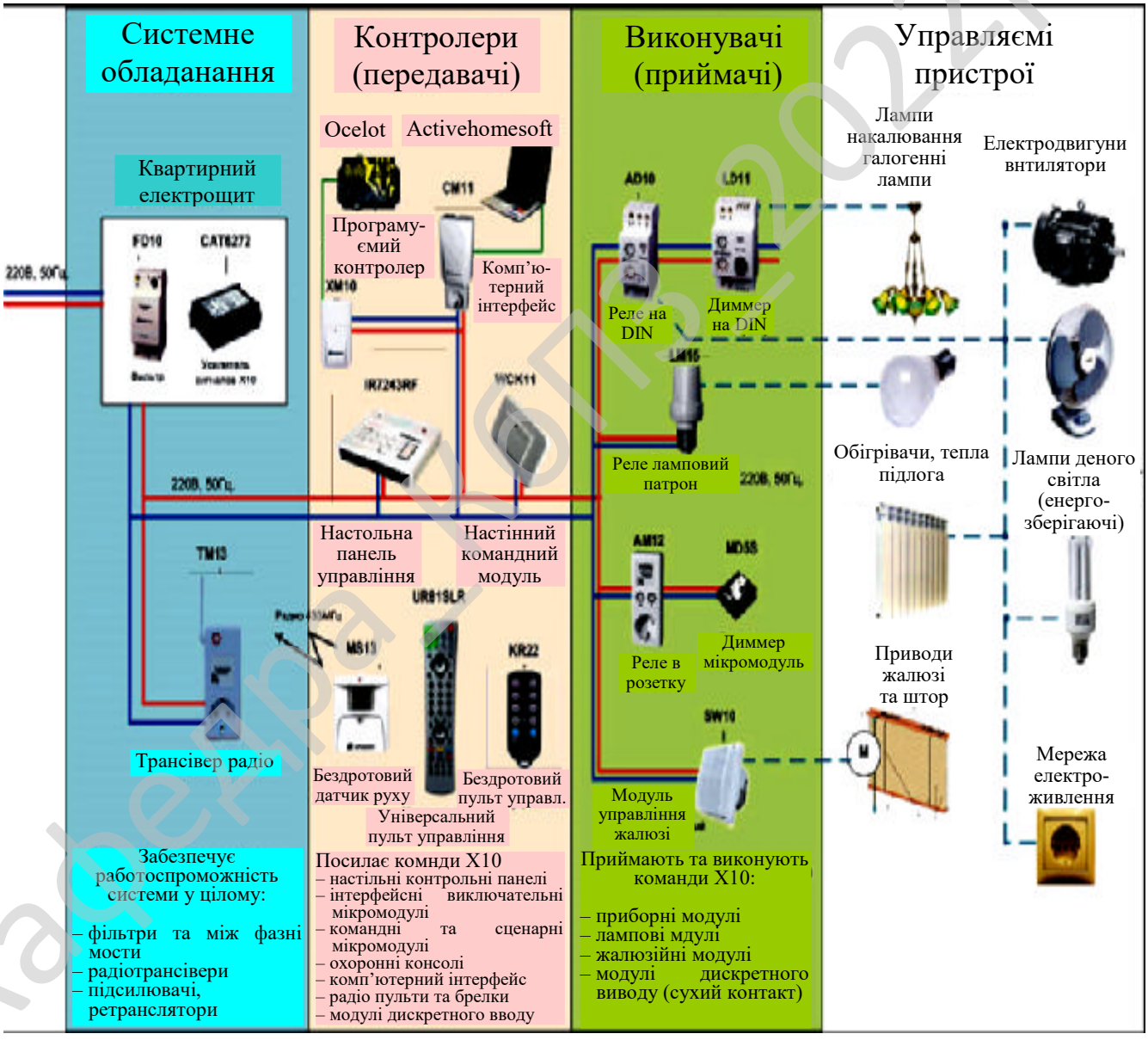


Рисунок 3.2 – Застосування устаткування X10

Рисунок 3.3 – Приклади пристроїв X10

Адреси X10

Кожний виконавчий пристрій X10 настроюється на певну адресу, що складається із двох 16-значних полів – Коду будинку (буквені значення A..P) і Коду пристрою (цифрові значення 1..16). Таким чином, усього можливі 256 адресні комбінації. Це максимально можлива кількість керованих груп пристроїв у мережі X10. Виконавчі пристрої X10 настроюються на певну адресу за допомогою двох механічних адресних селекторів-коліщат. Більшість контролерів X10 також вимагають попереднього налаштування адрес.

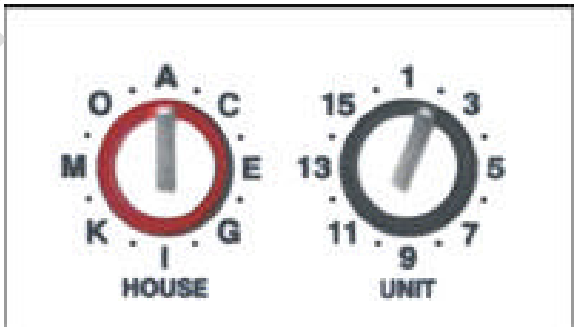


Рисунок 3.4 – Виконавчий пристрій X10

Для проходження команд контролер і виконавець повинні бути настроєні на однакові адреси.

Команди X10

У протоколі X10 передбачено шість базових команд:

– Включити (On).

– Виключити (Off).

– Яскравіше (Bright).

– Темніше (Dim).

– Включити все світло (ALL Ligts ON).

– Виключити всі (ALL Units OFF).

Сигнали X10

Технологія X10 використовує цифрове подання сигналів управління. Інформація кодується двійковим кодом і передається по електричній мережі за допомогою високочастотних імпульсів. Кожний переданий імпульс відповідає одному біту інформації c_j значенням "1". Передача чергового імпульсу відбувається в момент часу, коли сіткова напруга приймає нульове значення.

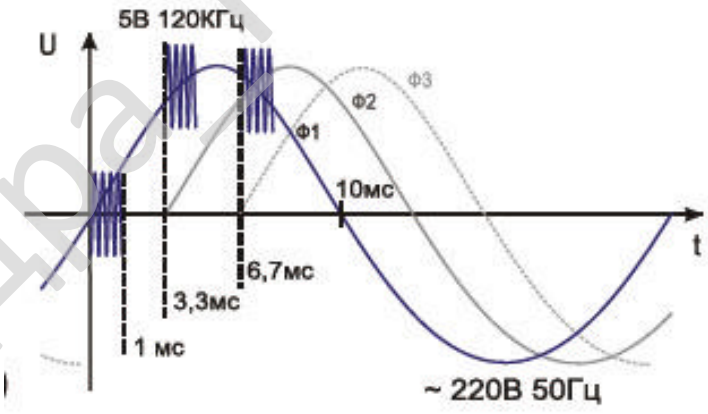


Рисунок 3.5 – Передача цифрових сигналів

Стандартна команда X10 передається протягом, приблизно, 50 періодів сіткової напруги частоти 50Гц або 1 сек. Більшість переданих по мережі X10 повідомлень містить, принаймні, два інформаційні поля – адреса пристрою,

якому ця команда адресована й властиво команду. Підключені до електромережі пристрої X10 приймають передані повідомлення, декодируют поле адреси одержувача й, якщо він збігається з їхньою власною адресою, виконують команду.

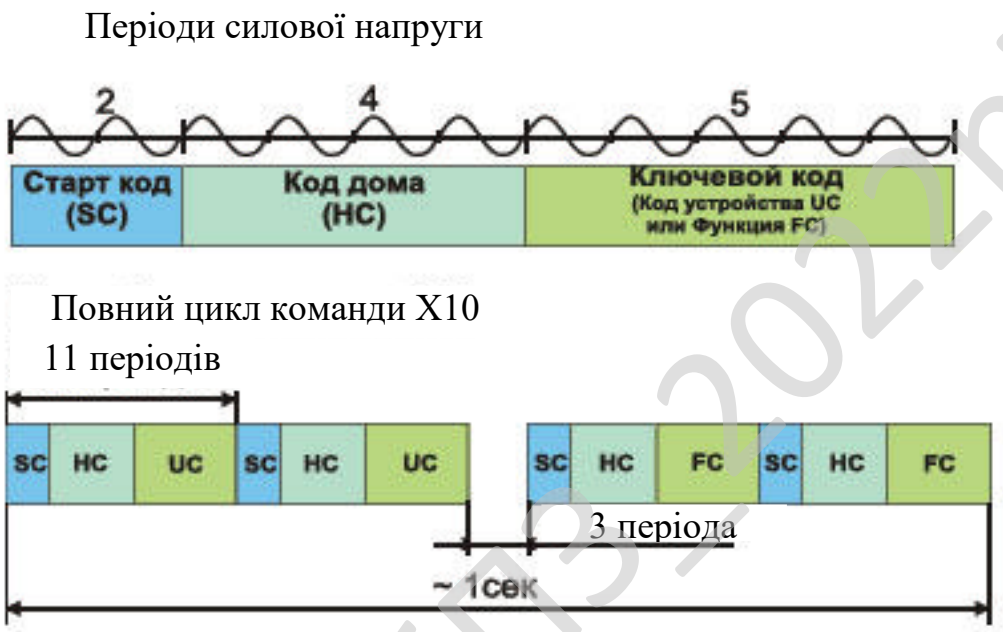


Рисунок 3.6 – Цикл виконання команди X10

Інтеграція X10

Основним засобом інтеграції мережі X10 із зовнішнім устаткуванням є PLC інтерфейс XM10. Будь-який контролер, що підтримує відкритий протокол обміну з XM10, може відправляти команди X10 в електромережу й, навпаки, одержувати з мережі інформацію про стан пристроїв X10. Так, наприклад, охоронні системи широко відомих у світі брендів – Ademco, DSC, Visonic і деяких інших – підтримують протокол X10 і дозволяють реалізувати інтегровані системи автоматизації й безпеки. Друга можливість апаратної інтеграції – підключення зовнішніх пристроїв по «сухому контакті». У лінійці X10 для цього використовуються модулі UM7206 і SM10.

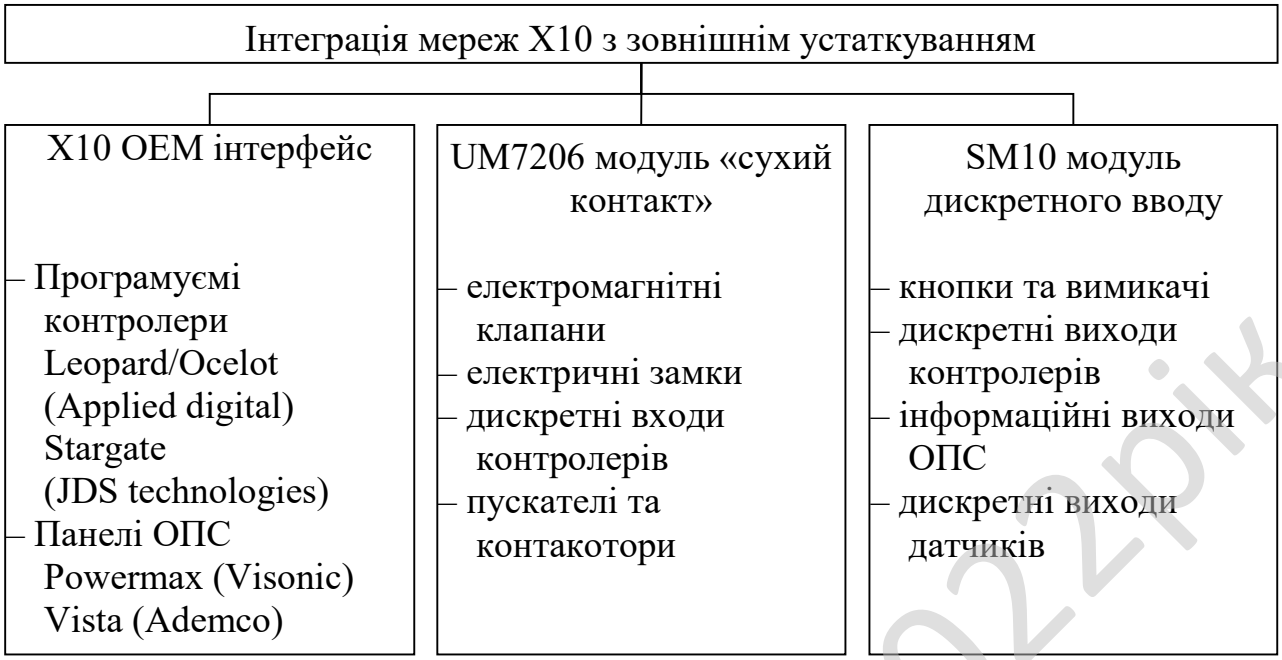


Рисунок 3.7 – Інтеграція мереж X10

Працездатність мережі X10

При установці автоматики X10 необхідно звертати особливу увагу на:

– перешкодозахищеність системи;

– забезпечення міжфазного проходження сигналу X10 при багатофазній схемі електроживлення;

– захист електричних кіл і пристроїв X10 від перенапруги, перевантаження й струмів короткого замикання.

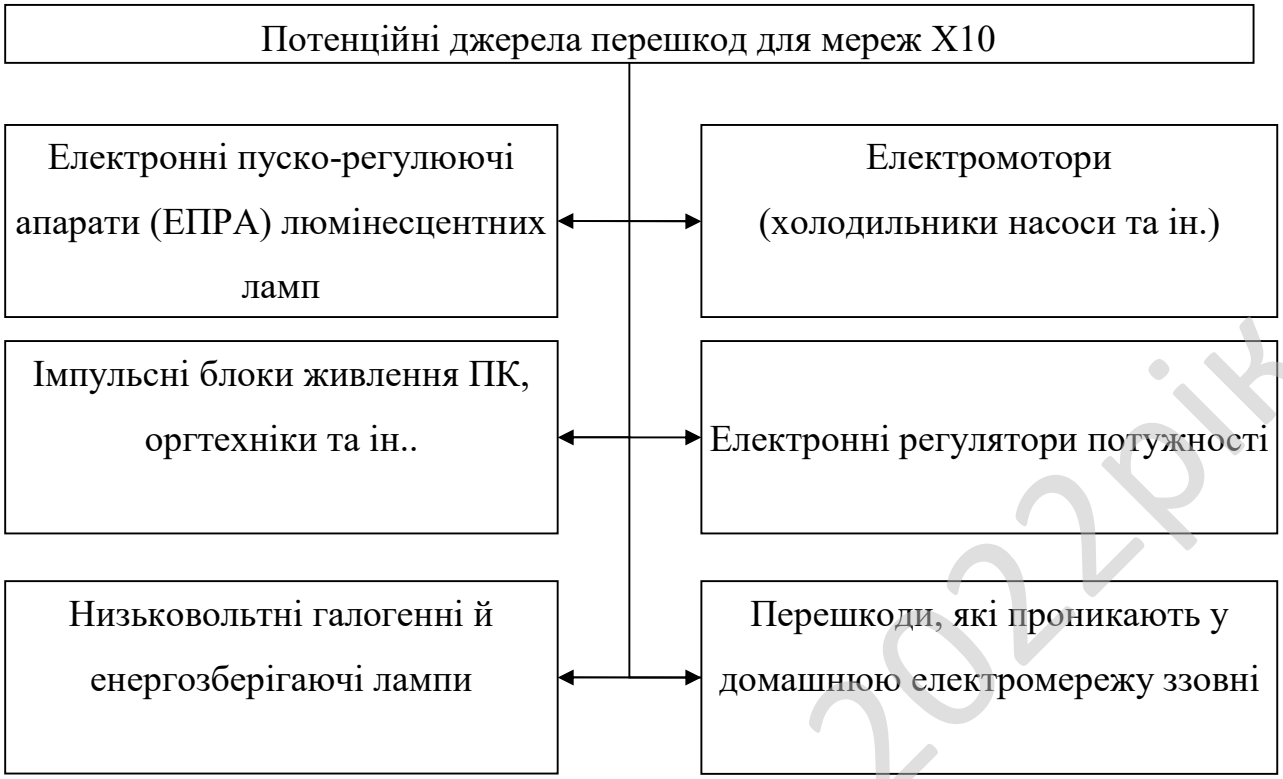


Рисунок 3.8 – Джерела перешкод для Х10

Міри, що рекомендуються, для усунення перешкод у мережах Х10:

– використання мережних фільтрів (FM10, FD10, TF678 і ін.) на введенні домашньої електромережі й у місцях підключення до електроживлення приладів-джерел перешкод;

– підключення пристроїв Х10 і приладів-джерел перешкод до різних фаз у багатофазній електромережі;

– заміна «шумливих» електроприладів на більш якісні моделі.

У цілому деякі електроприлади стають джерелами перешкод для пристроїв Х10, і при дотриманні нескладних правил захисту від високочастотного шуму автоматика Х10 працює надійно протягом довгого часу.

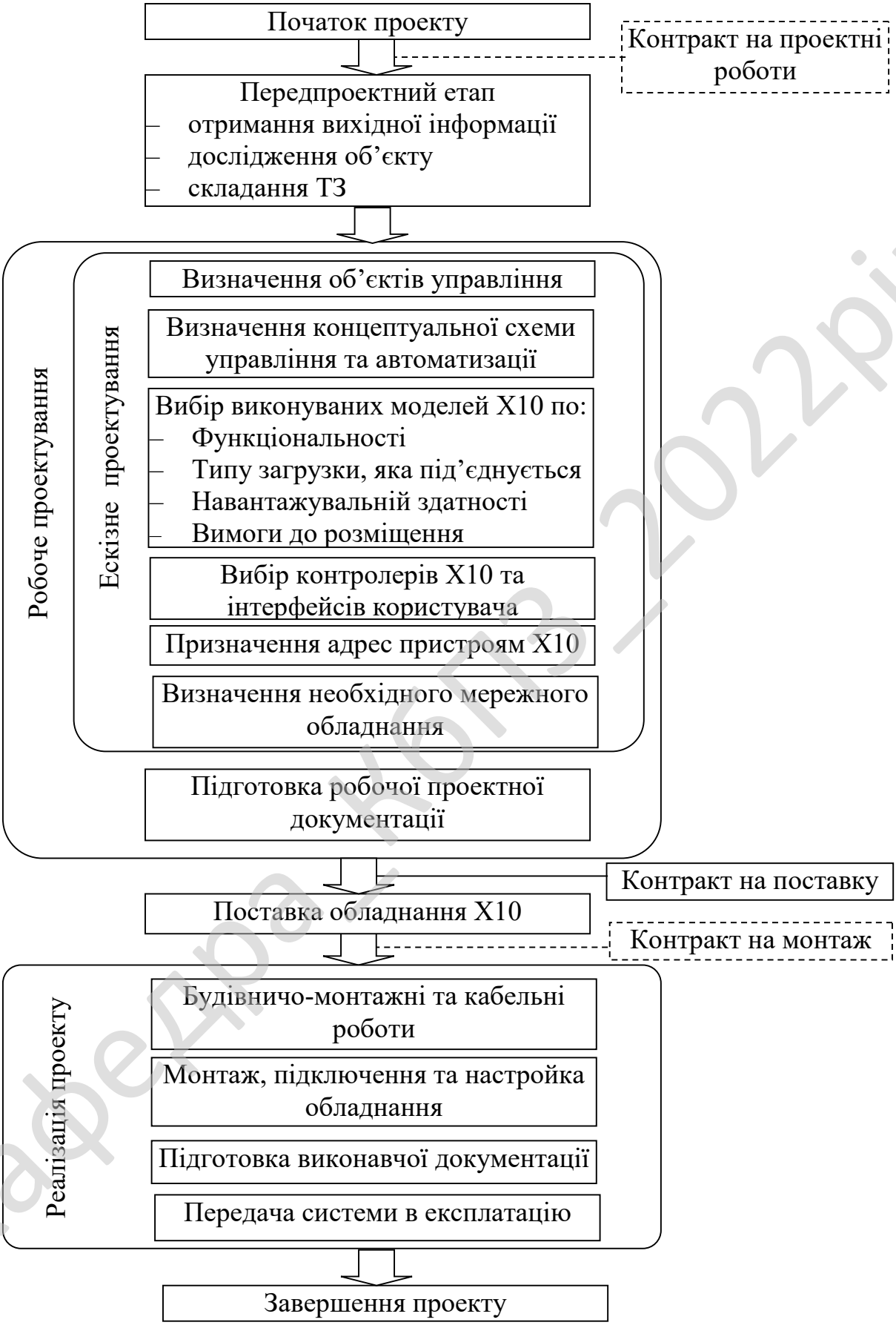


Рисунок 3.10 – Виконання проекту X10

Проект X10

Вихідною інформацією для проектів X10, як і для більшості проектів домашньої автоматизації, є:

– плани приміщень

– однолінійна електрична схема

– плани розміщення електроустаткування – світильників, світлових вимикачів, електричних розеток, точок прямого підключення електроустаткування, електричних щитків і шаф, комутаційних вузлів

– перелік автоматизуємих групових і одиночних електричних навантажень, із вказівкою їхніх типів і потужності

– плани існуючої кабельної проводки й кабельний журнал

– відомості про стан об'єкта й можливості проведення кабельних робіт

– вимоги замовника до функціональності проектованої системи управління

При виконанні проекту X10 рекомендується технологічна послідовність робіт зображена на рисунку 3.10.

3.2 Розробка структурної схеми

На рисунку 3.11 зображена узагальнена структурна схема інтелектуального дому.

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

– Система безпеки.

– Система комфорту.

– Інформаційна система.

– Система диспетчеризації.

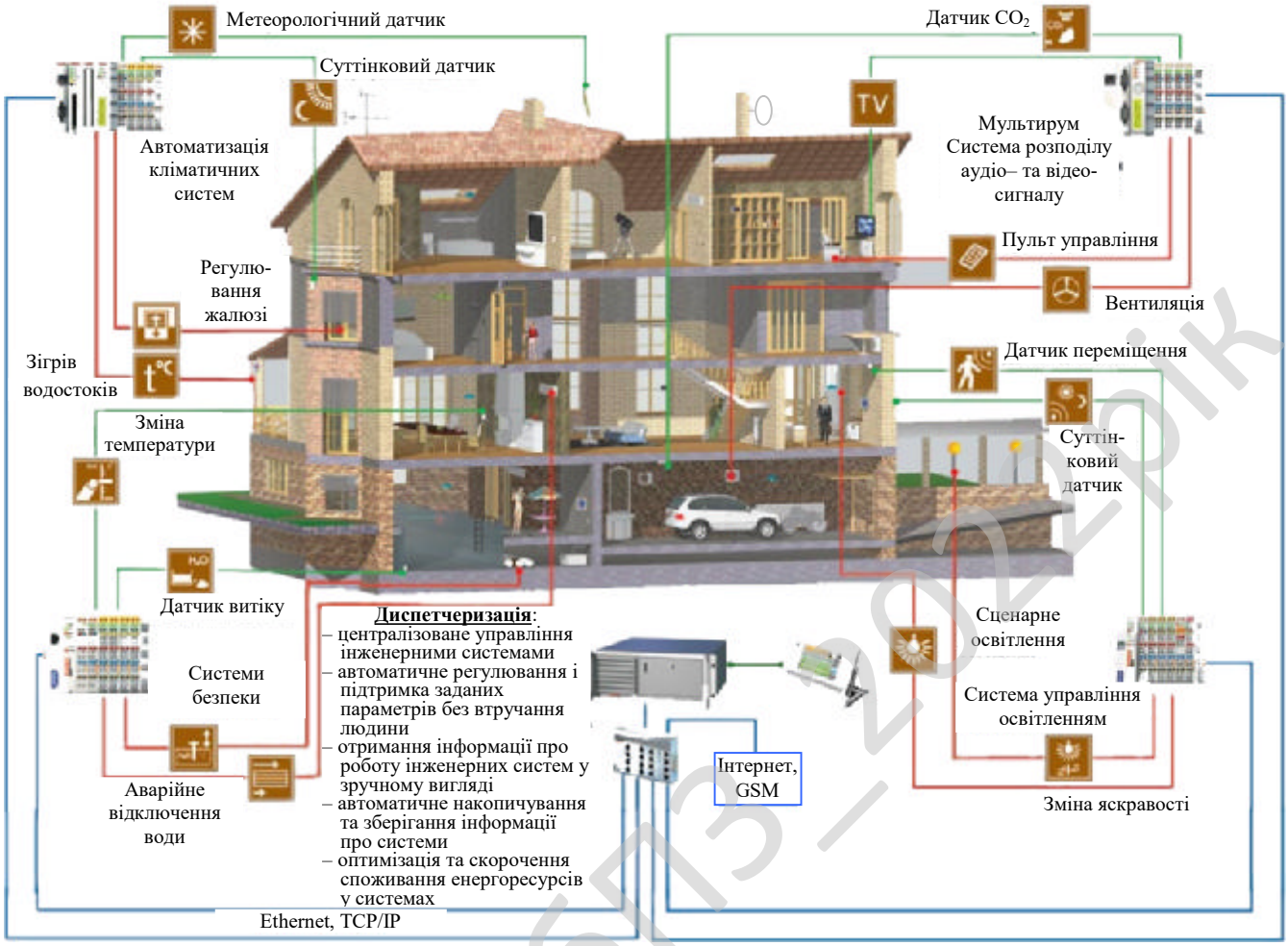


Рисунок 3.11 – Структурна схема інтелектуального дому

Розглянемо які підсистеми входять у перераховані вище системи.

Система безпеки:

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

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

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

Система комфорту:

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

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

– Система "Домашній кінотеатр".

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

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

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

– Управління системою вентиляції й кондиціювання.

– Управління системою опалення.

– Управління сауною, басейном.

Інформаційна система:

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

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

– Віддалене управління всіма системами будинку через Інтернет.

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

Система диспетчеризації:

– Система безперебійного електропостачання

– Управління системою опалення, казаном водонагрівача

– Контроль витоку води, газу.

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

На рисунку 3.12 відображено структурну схему взаємодії систем інтелектуального будинку

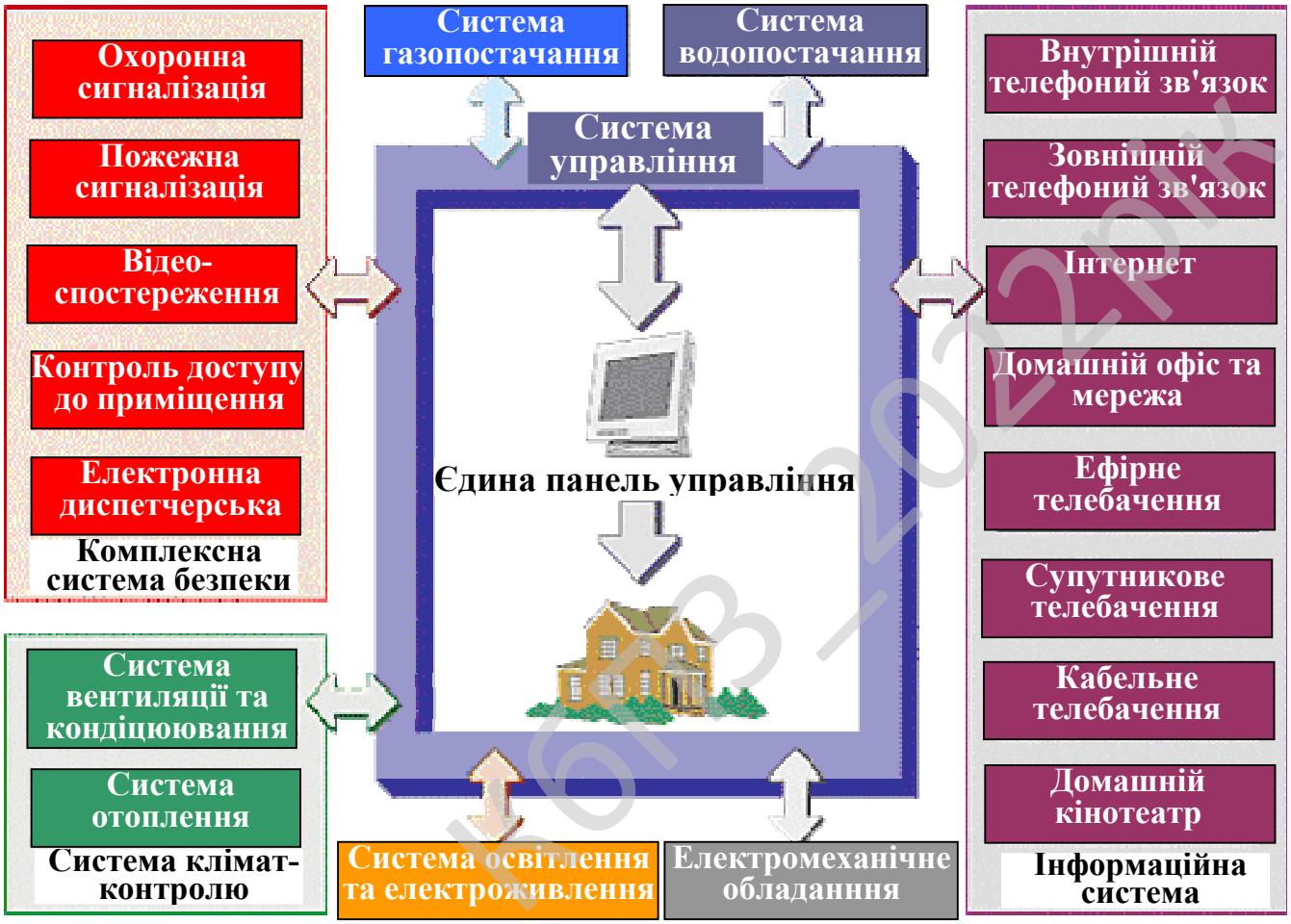


Рисунок 3.12 – Структурна схема взаємодії систем інтелектуального будинку

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

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

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

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

Основні складові інтелектуального будинку

Зв'язок:

- телефонний зв'язок внутрішня (інтерком) і локальні АТС;
- телефонний зв'язок зовнішня провідна, радіорелейна, супутникова;
- інтернет – телефонія;
- системи відеоконференцій.

Мультимедіа:

- наземне й кабельне телебачення;
- супутникове телебачення;
- домашнє відео;
- домашній кінотеатр;
- мульти– аудіо й відео (multiroom).
- один пульт управління для всіх систем

Інформаційні системи:

- локальна комп'ютерна мережа;
- домашній офіс;
- широкополосний доступ у глобальну мережу (Інтернет).

Безпека:

- пожежна сигналізація й система автоматичного пожежогасіння;

– охоронна сигналізація;

– система контролю доступу;

– система зовнішнього й внутрішнього відео спостереження;

– система управління паркінгом;

– система внутрішнього оповіщення (радіомережа);

– аварійний контроль інженерних систем;

– система екологічного контролю.

Водопостачання й газопостачання:

– автоматичне наповнення ванн, басейнів і накопичувальних резервуарів;

– автономне й резервне нагрівання води;

– запобігання витоку водопроводу й витоку газу;

– управління ландшафтними водяними системами (фонтани, водоспади);

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

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

Система електроживлення:

– безперебійне й гарантоване електропостачання;

– захист від поразки електричним струмом людей і тварин;

– автоматизація управління електроживленням побутових приладів;

– запобігання перевантажень і короткого замикання в електричній мережі;

– управління якістю електроживлення – моніторинг, стабілізація й фільтрація;

– облік витрат й управління споживанням електроенергії.

Освітлення внутрішнє й зовнішнє:

– аварійне й чергове освітлення;

– автоматизація управління світлом;

– гнучке налаштування світлових груп;

– дистанційне й віддалене управління світлом;

– програмування світлових сцен.

Опалення, вентиляція, кондиціювання:

– автоматизація управління температурою повітря;

– контроль якості повітря;

– узгодження роботи різних кліматичних систем;

– облік витрати й управління споживанням теплової енергії.

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

3.3 Розробка функціональної схеми

Функціональна структура розробленої мережі складвється з наступних болків, як взаємодіють з блоком функцій X10:

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

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

Рисунок 3.13 – Функціональна схема системи

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

– управління з екрану ЕОМ;

– релейне управління;

– діммування;

– «проходні» схеми вимикання;

– бездротове управління електрикою;

– для аудіо/відео та світла один пульт ДУ;

– віддалене управління за телефоном.

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

– бездротове аудіо/відео;

– універсальні навчаємі пульти ІЧ ДУ;

– радіотрансляція ІЧ ДУ;

– бездротове аудіо/відео спостереження;

– радіоуправління ПЗ ЕОМ.

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

– звукова тривожна сигналізація;

– охоронна сигналізація;

– світлова тривожна сигналізація;

– тривожне оповіщення по телефону;

– світлова імітація наявності людей;

– функція «Паніка».

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

– макроси та сценарії;

– за температурою повітря;

– по руху людини;

– за рівнем освітленості;

– по сухому контакту;

– за часом доби.

3.4 Розробка діаграми процесів

На рисунку 3.14 зображена діагрпмп взаємодії процесів системи. З неї бачимо, що після ввімкненя системи запускаються два основних процеси:

– процес надсилання керуючих слів на пристрої будинку;

– процес читання показів пристроїв.

Розглянемо роботу цих процесів більш детальніше.

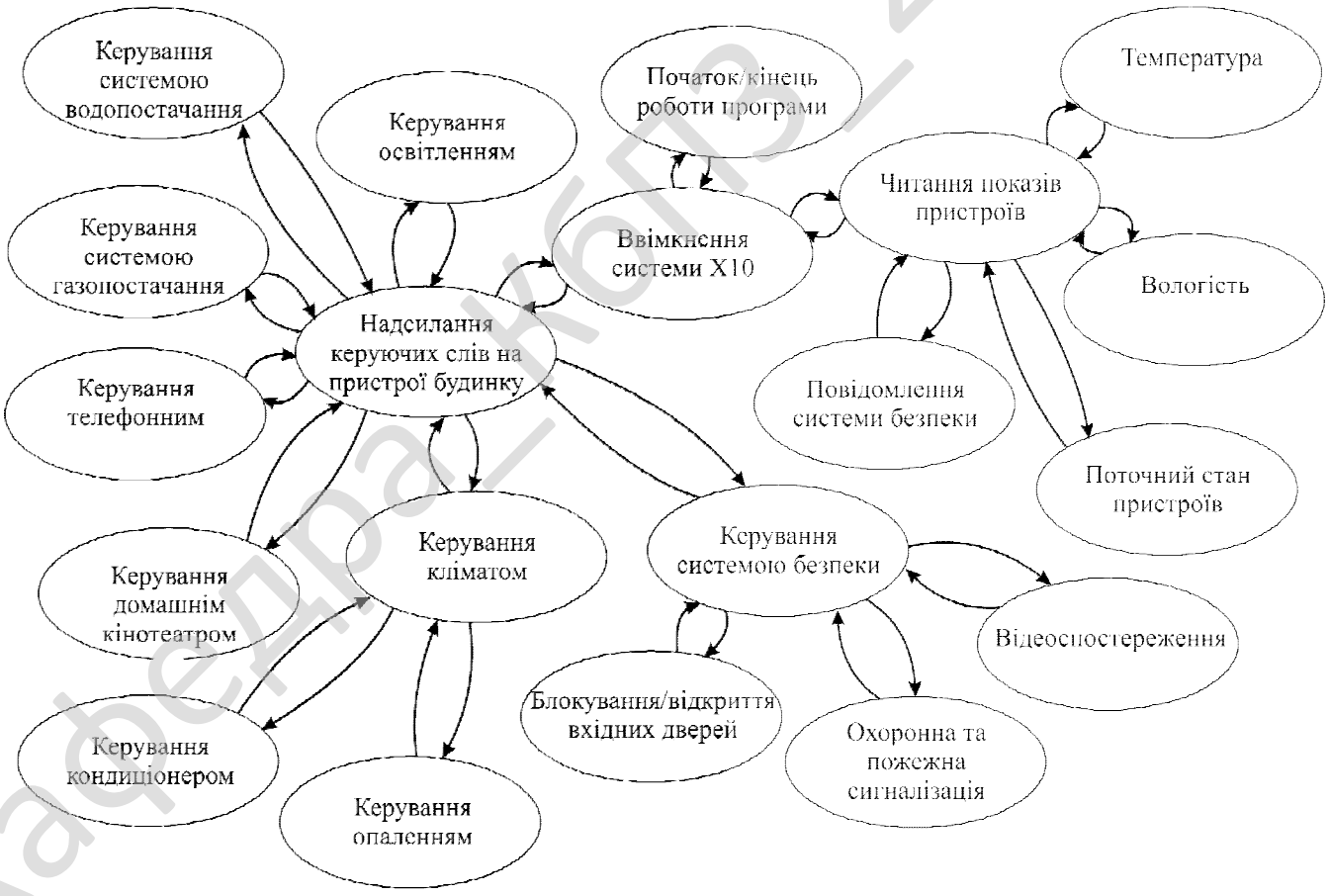


Рисунок 3.14 – Діаграма процесів системи

Почнемо з процесу читання показів пристроїв. Цей процес працює з наступними процесами:

– процес читання показів температури у приміщені та за межами дому;

– процес читання показів вологості у приміщені та за межами дому;

– процес читання показів поточного стану приборів;

– процес читання повідомлень системи безпеки.

Розглянувши взаємодію процесів зв'язаних з процесом читання показів пристроїв, перейдемо до розгляду процесу надсилання керуючих слів на пристрої будинку.

Ц цим процесом взаємодіють наступні процеси:

– процес керування освітленням;

– процес керування системою водопостачання;

– процес керування системою газопостачання;

– процес керування телефоном;

– процес керування домашнім кінотеатром;

– процес керування кліматом;

– процес керування системою безпеки.

Процес керування кліматом взаємодіє з наступними процесами:

– процес керування кондиціонером;

– процес керування опаленням.

Процес керування системою безпеки взаємодіє з наступними процесами:

– процес блокування/відкриття вхідних дверей;

– процес охоронна та пожежна сигналізація;

– процес відеоспостереження.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема основної програми, розробленої у результаті виконання магістерського проектування.

Після завантаження програмного продукту відбувається виведення головного вікна на екран. У цьому вікні користувач може вибрати той або інший розділ, з яким він буде працювати.

Користувач може працювати з наступними програмними блоками:

– управління освітленням;

– керування водопостачанням;

– керування газопостачанням;

– керування кліматом;

– керування телефоним зв'язком;

–керування домашнім кінотеатром;

– надання довідки.

Якщо користувач обирає роботу з програмним блоком управління освітленням, то відбуваються наступні дії.

Спершу визначається поточний стан освітлення, цей поточний стан виводиться на монітор, для візуалізації процесу прийняття рішення користувачем.

Після цього відбувається керування освітленням у ручному, або автоматичному режимі, в залежності від того, як це вирішить користувач.

При керуванні освітленням крім того. Що визначається де саме будуть горіти лампи, задається ще й яскравість їх горіння.

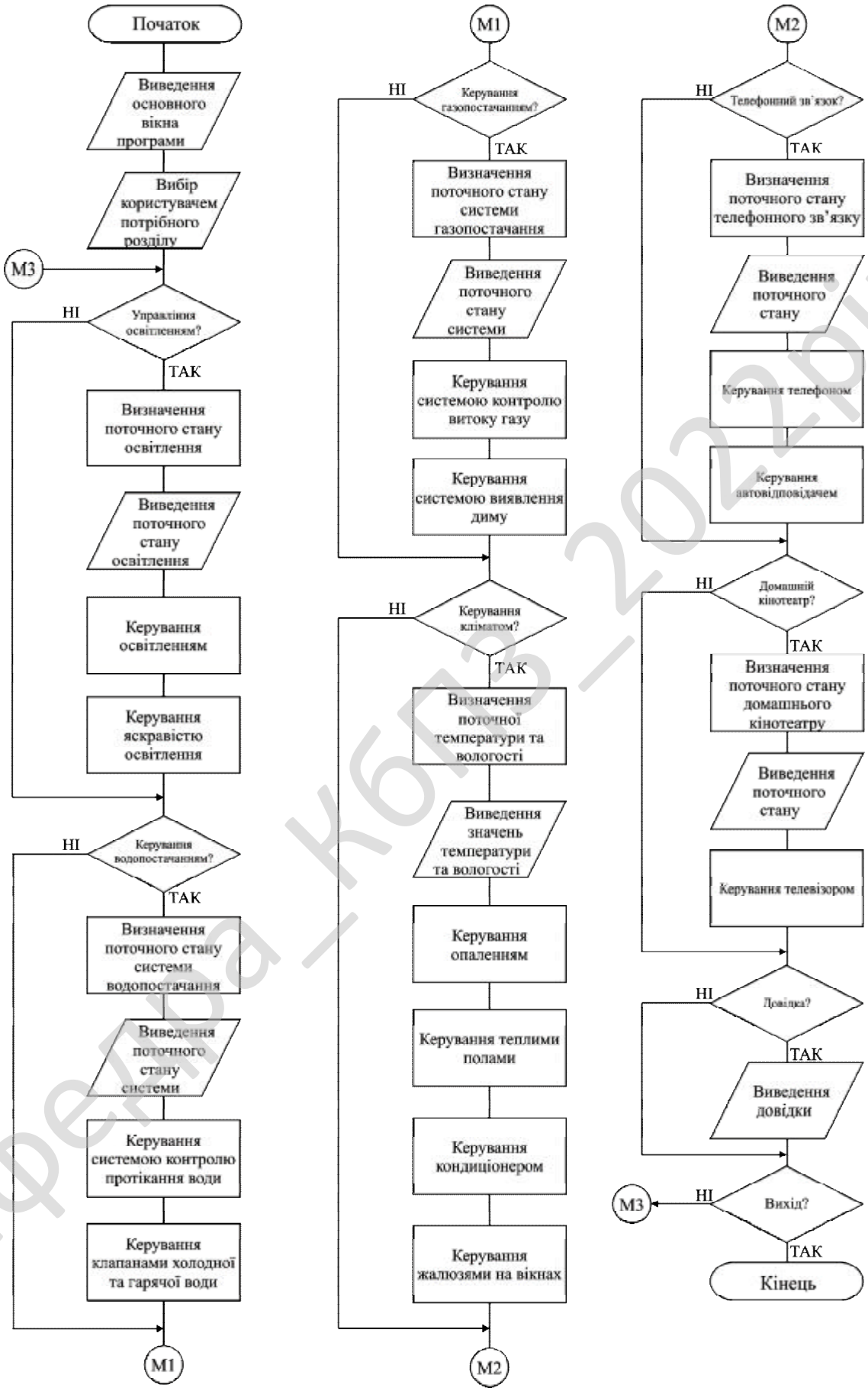


Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає роботу з програмним блоком керування водопосточанням, то відбуваються наступні дії.

Спершу визначається поточний стан системи водопостачання та виводиться на екран.

Після цього користувач здійснює керування системою контролю протікання води та клапанами холодної й гарячої води, у ручному або автоматичному режимі.

Якщо користувач обирає роботу з програмним блоком керування газопостачанням, то відбуваються наступні дії. Спершу визначається поточний стан системи газопосточання який виводиться на екран.

Після цього у автоматичному режимі здійснюється керування системою контролю витоку газу та системою виявлення диму.

Якщо користувач обирає роботу з програмним блоком керування кліматом, то відбуваються наступні дії.

Спершу визначається поточна температура та вологість.

Після цього в залежності від показників датчиків відбувається управління наступними системами:

– опалення;

– теплі поли;

– кондіціонером;

– жалюзі на вікнах.

Якщо користувач обирає роботу з програмним блоком керування телефонного зв’язку, то відбуваються наступні дії.

Визначається поточний стан телефонного зв’язку, який виводиться на екран. Після того як він визначився відбувається управління телефоном та автовідповідачем.

Якщо користувач обирає роботу з програмним блоком керування домашнім кінотеатром, то відбуваються наступні дії.

Визначається поточний стан кінотеатру, який виводиться на екран. Після

цього користувач приймає рішення по управлінню відео- та аудіо- апаратурі.

При входженні у меню довідки, користувач отримує довідку про програмку, та режими її функціонування й експлуатації.

Якщо користувач натискає на кнопку вихід, то програма закінчує свою роботу.

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

На рисунку 4.2 зображена блок-схема підпрограми керування пристроями інтелектуального будинку. З неї ми бачимо, що управління пристроями відбувається у наступній послідовності.

Спершу вказується код конкретного дому.

Після цього задається код пристрою та код програми.

Вибір пристрою здійснюється передачею керуючого слова у котрому вказані стартовий код, код дому та код пристрою.

Це керуюче слово передається вибраному пристрою.

Пристрій визначає чи є команда, яку він отримав командою керування.

Якщо ні, то він переходить до розпізнавання її як команди запиту стану.

Якщо він визначає, що отримав команду керування, то починає чекати підтвердження.

Якщо підтвердження отримане. То команда виконується, у іншому випадку команда не виконується.

Після читання команди керування читається команда запиту стану.

Якщо вона не прочиталась, то видається повідомлення про невірний формат команди й підпрограма закінчує свою роботу.

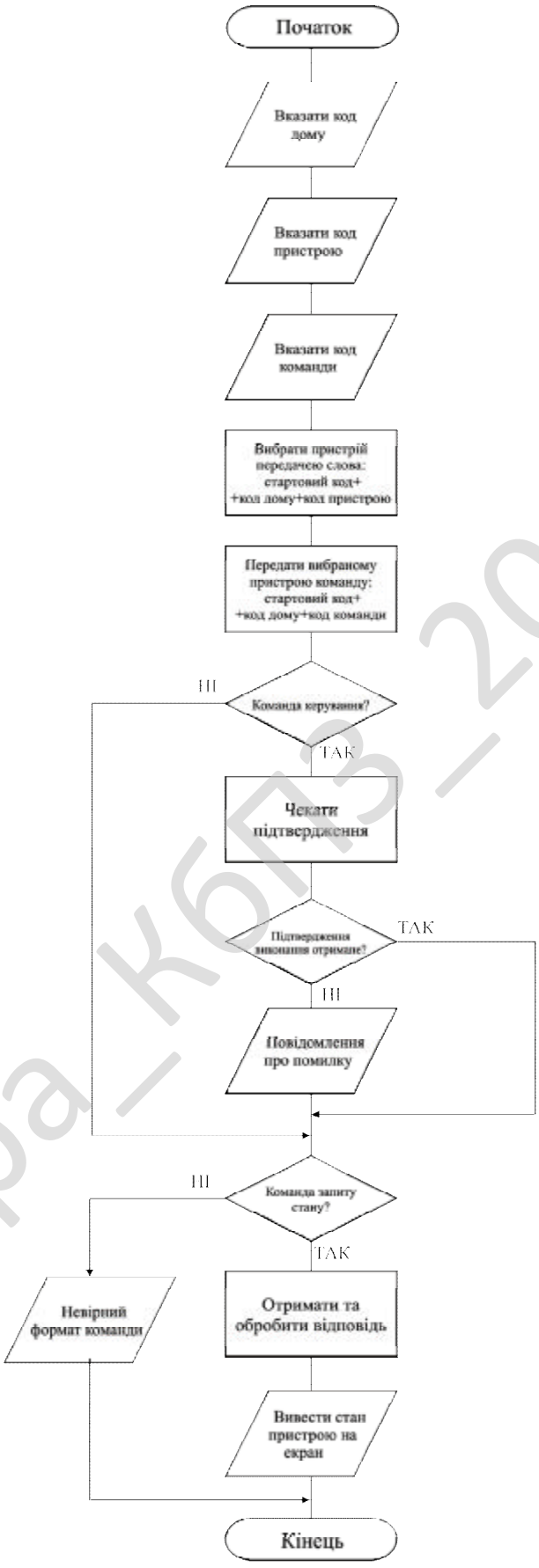


Рисунок 4.2 – Блок-схема роботи підпрограми керування пристроями інтелектуального будинку

У випадку ж, якщо команда запиту стану була прочитана, то пристроєм передається відповідь, які програма отримує та обробляє. Результати обробки відповіді на команду стану виводяться на екран.

На цьому підпрограма керування пристроями закінчує свою роботу.

Хоча сигнали управління X10 передаються й приймаються безпосередньо по електричній мережі, для під'єднання пристроїв до комп'ютера можна використовувати різні топології, такі як наприклад Ethernet, RS-232, RS-485 та інші. Для вирішення поставленого у магістерському проекті завдання, було обрано інтерфейс Ethernet.

Стандарт X10 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 X10 може бути об'єднане до 256 груп пристроїв з різними адресами.

З погляду логіки організації внутрімережної взаємодії всі пристрої X10 можна розбити на дві більші групи: контролери й виконавчі модулі.

Контролери відповідають за генерацію команд X10 і, крім ручного кнопкового керування, можуть мати убудований таймер або спеціалізований пристрій уведення зовнішнього впливу (датчик освітленості, фотоприймач інфрачервоного випромінювання від пульта дистанційного керування й т.д.).

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

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

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

Для передачі команди 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:

11100101101001010110011111001011010010101100100000011100101101001011001101110010110100101100110.

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

Команди X10

У протоколі X10 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 X10

Технологія X10 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 X10 передається протягом, приблизно, 50 періодів мережної напруги частоти 50Гц або 1 сек.

Більшість переданих по мережі X10 повідомлень містить, принаймні, два інформаційні поля: адреса пристрою, якому ця команда адресована, і властиво команду. Підключені до електромережі пристрої X10 приймають передані повідомлення, декодують поле адреси й, якщо він збігається з їх власним, виконують команду.

Інтеграція X10

Основним засобом інтеграції мережі X10 із зовнішнім устаткуванням є PLC інтерфейс XM10. Будь-який контролер, що підтримує відкритий протокол обміну з XM10, може відправляти команди X10 в електромережу й, навпаки, одержувати з мережі інформацію про стан пристроїв X10.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer. Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES. В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду. В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S-блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст. Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході). На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не

використовується. У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел. Довжина блоку завжди вибирається рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

– Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.

– Переміщення, за допомогою якого біти повідомлення переупорядковуються.

– Операція додавання по модулю 2, позначувана XOR або ⊕.

– Операція додавання по модулю 2^{32} або по модулю 2^{16}.

– Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K. Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

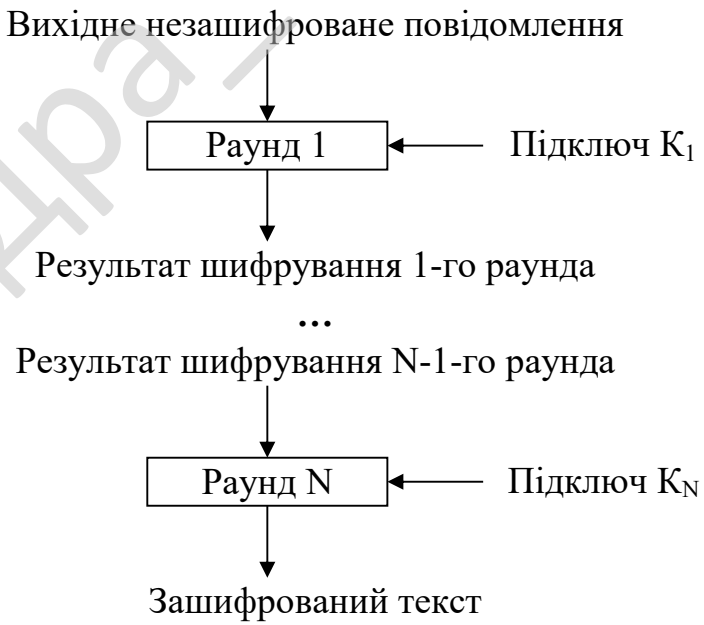


Рисунок 4.3 – Структура алгоритму алгоритмі Lucifer

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення призначене для управління smart home з використанням протоколу X10.

Програмно-апаратні вимоги до комп'ютера:

- Загальний обсяг ОЗП: 512 Мбайт.
- Вільний простір на жорсткому диску: 23 Мбайти.
- Операційна система Microsoft Windows 10/11.
- Інтерфейс ethernet.

Перелік необхідного обладнання:

Обов'язкове обладнання:

- Мережні пристрої – для об'єднання всіх інших пристроїв з комп'ютером (вита пара, комутатори, концентратори тощо).
- Передавачі – дозволяють передавати спеціальні коди команд у форматі X10 по електромережі. Такими пристроями є: **програмуємі таймери**, що посилають сигнали в потрібний час; **комп'ютерні модулі**, що виконують задані програми по керуванню електроприладами; **датчики** температури, освітленості, руху й ін., які при настанні певних подій посилають відповідні сигнали приймачам.
- Приймачі – приймають команди X10 і виконують їх: включають або виключають світло, регулюють освітленість і т.д. На кожному приймачі є селектори установки його адреси: 16 можливих кодів будинку (A – P) і 16 можливих кодів модуля (1 – 16), тобто всього 256 різних адрес. Кілька приймачів можуть мати однакову адресу, у цьому випадку вони управляються одночасно.
- Вимірювальне устаткування – використовується для вимірювання рівнів корисних сигналів X10 і перешкод в електромережі при виконанні монтажних і пусково-налагоджувальних робіт.

Рекомендоване обладнання:

– Лінійне устаткування – повторювачі/ретранслятори сигналів, фільтри стрибків напруги або струму, протишумові фільтри, блокатори сигналів. Ці пристрої використовуються для підвищення надійності й безвідмовності системи в цілому. Хоча в простих системах можливе досягнення прекрасних результатів і без використання цих засобів, але завжди краще підстраховатися.

– Трансивери – приймають сигнали від інфрачервоних або радіо-пультів дистанційного керування й передають їх в електромережу, перетворивши у формат X10.

– Пульти ДУ – забезпечують дистанційне керування пристроями X10 по ІЧ або радіо-каналам.

Функціональні можливості програми:

1. Управління освітленням будинку:
   – вмикання/вимикання ламп;
   – регулювання яскравості світла від ламп;
   – перегляд поточного стану ламп;
   – імітація присутності господарів, шляхом почергового вмикання вимикання ламп у різних кімнатах (для відлякування крадіїв).
2. Управління водопостачанням:
   – система контролю протікання води;
   – відкриття/закриття клапанів холодної та гарячої води.
3. Управління газопостачанням:
   – система контролю витоку газу;
   – система виявлення диму.
4. Управління кліматом будинку:
   – перегляд поточної температури та вологості у будинку;
   – управляння опаленням (вмикання/вимикання, регулювання рівня температури, автоматична робота у вказаному діапазоні часу);
   – управління гарячими полами (вмикання/вимикання, регулювання

рівня температури);

– управління кондиціонером (вмикання/вимикання, регулювання рівня температури та вологості);
– управляння жалюзями на вікнах.

5. Управління телефонним зв'язком:
   – управління телефоном (вмикання/вимикання, регулювання гучності, набір номера);
   – управління автовідповідачем (вмикання/вимикання, гучність).
6. Управління домашнім кінотеатром:
   – вмикання/вимикання телевізора;
   – регулювання гучності;
   – вибір телеканалу.
7. Управління системою безпеки будинку:
   – управління охоронною сигналізацією;
   – управління пожежною сигналізацією;
   – управління системою відеоспостереження;
   – блокування/відкриття вхідних дверей.

Головне вікно програми зображене на рисунку 5.1.

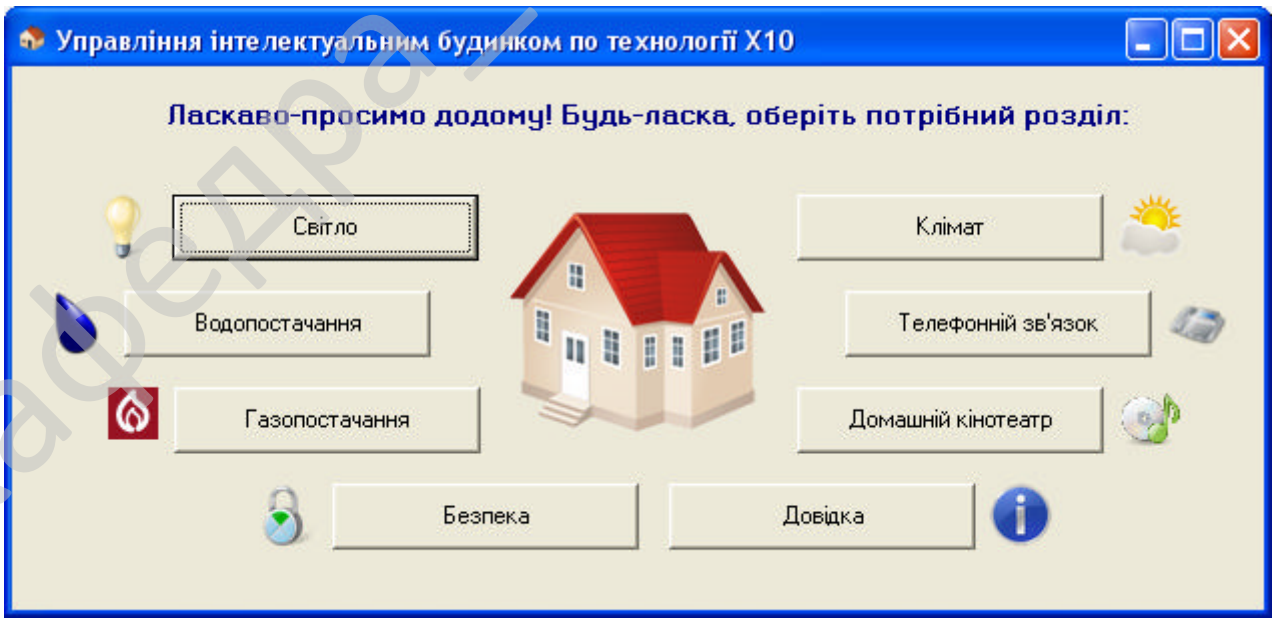


Рисунок 5.1 – Головне вікно програми

З рисунку видно, що за допомогою головного вікна користувач здійснює вибір розділу програми, в який він хоче перейти, а саме:

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

Вікно управління освітленням зображене на рисунку 5.2.

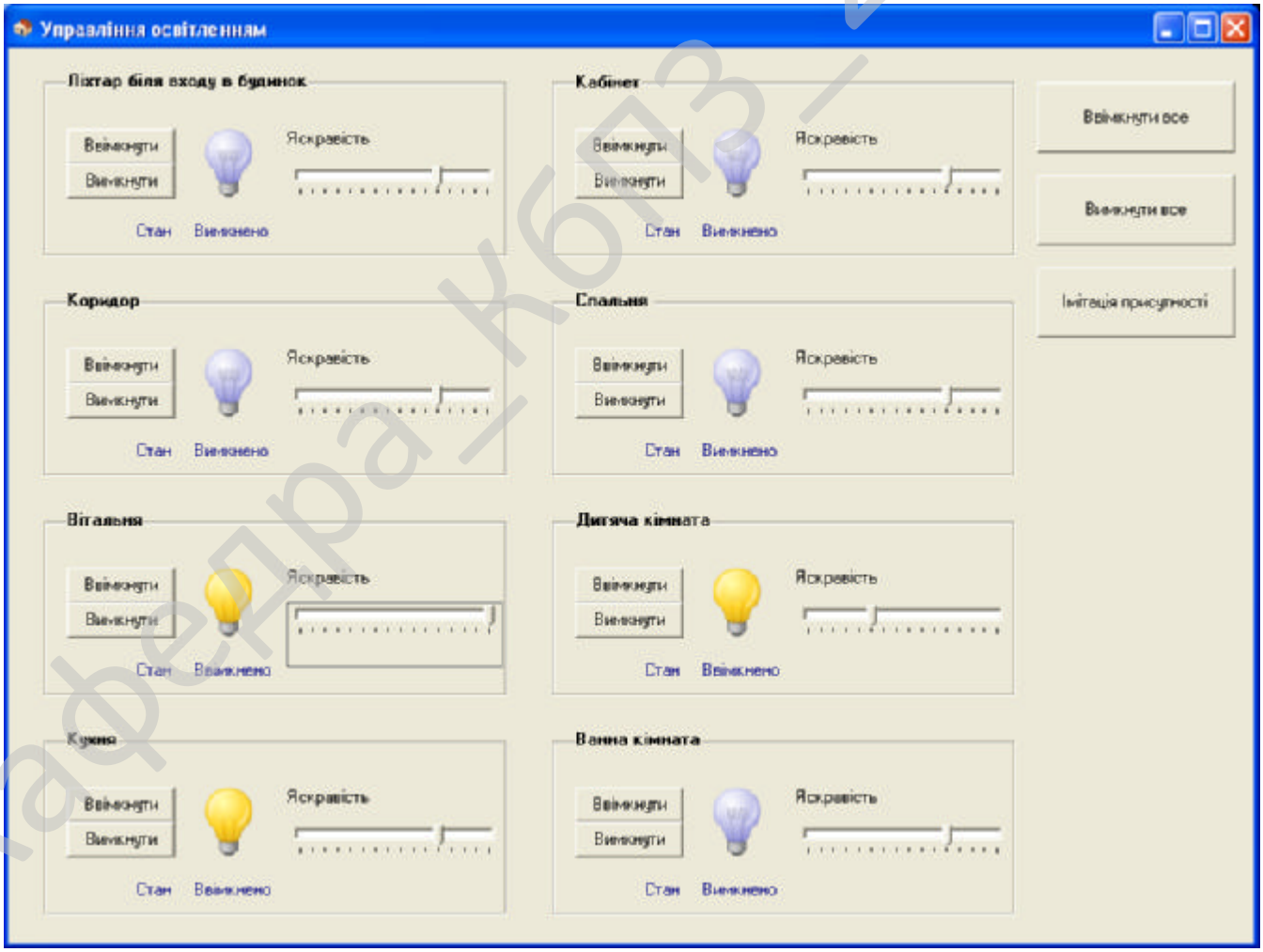


Рисунок 5.2 – Управляння освітленням

Вікно управління водопостачанням зображене на рисунку 5.3.

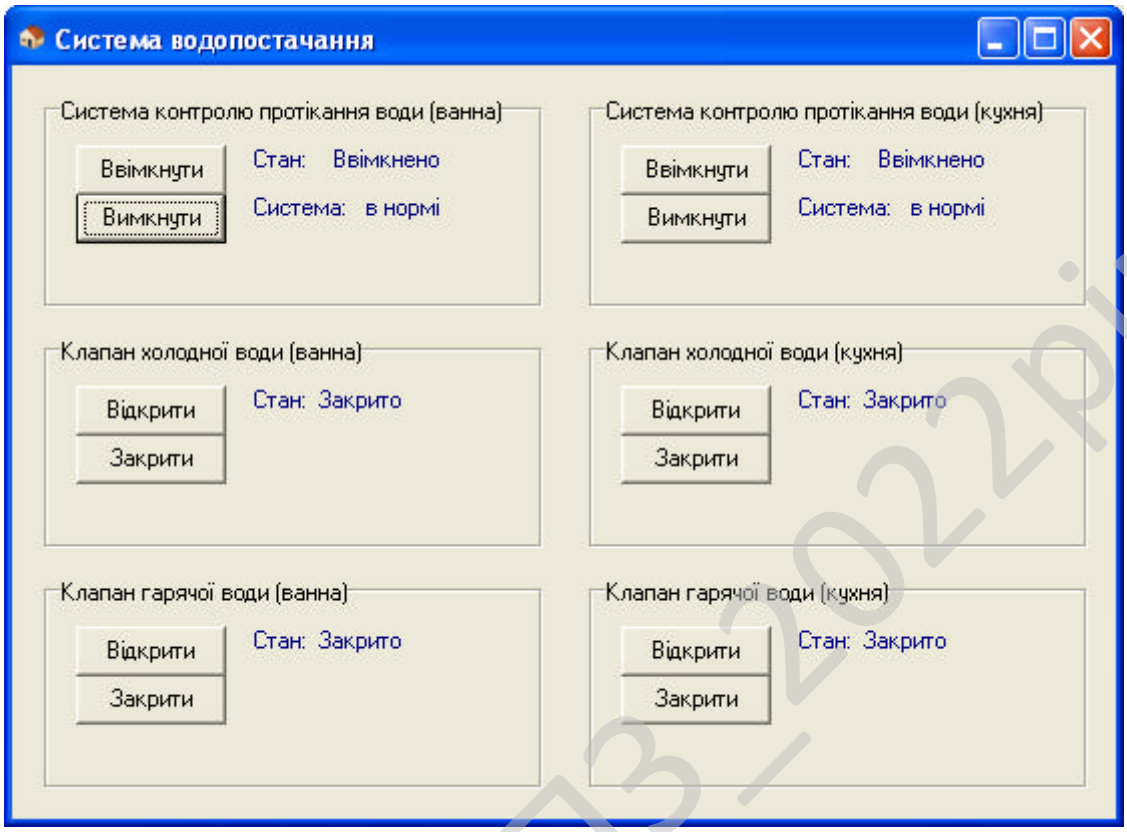


Рисунок 5.3 – Управляння водопостачанням

Вікно управління газопостачанням зобрежене на рисунку 5.i4.

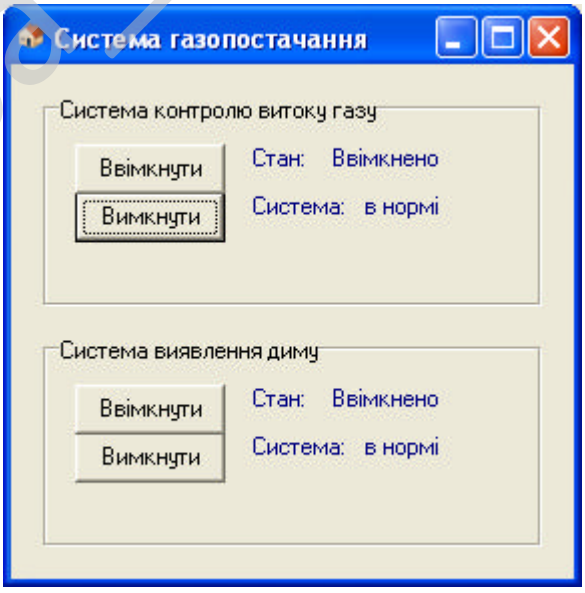


Рисунок 5.4 – Управляння газопостачанням

Вікно управління системою безпеки будинку зображене на рисунку 5.5.

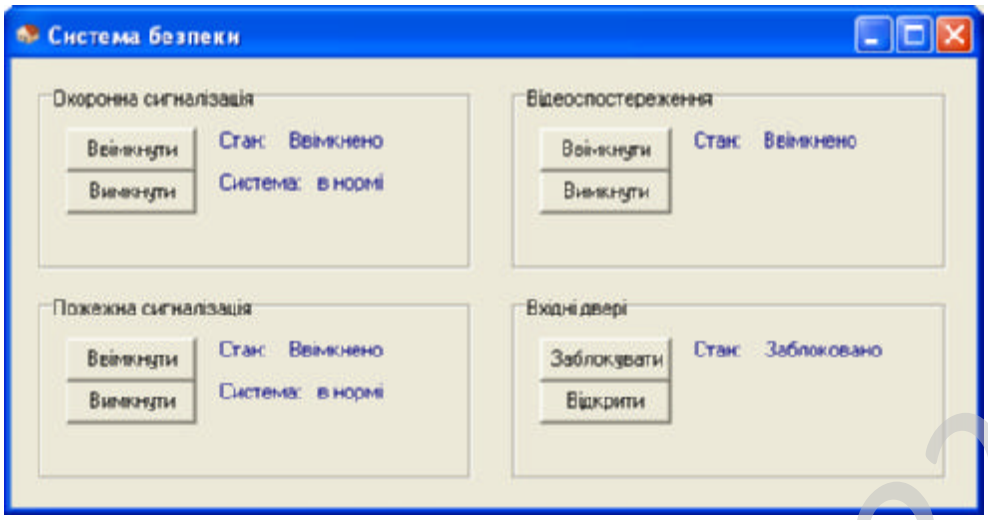


Рисунок 5.5 – Система безпеки будинку

Вікно управління кліматом будинку зображене на рисунку 5.6.

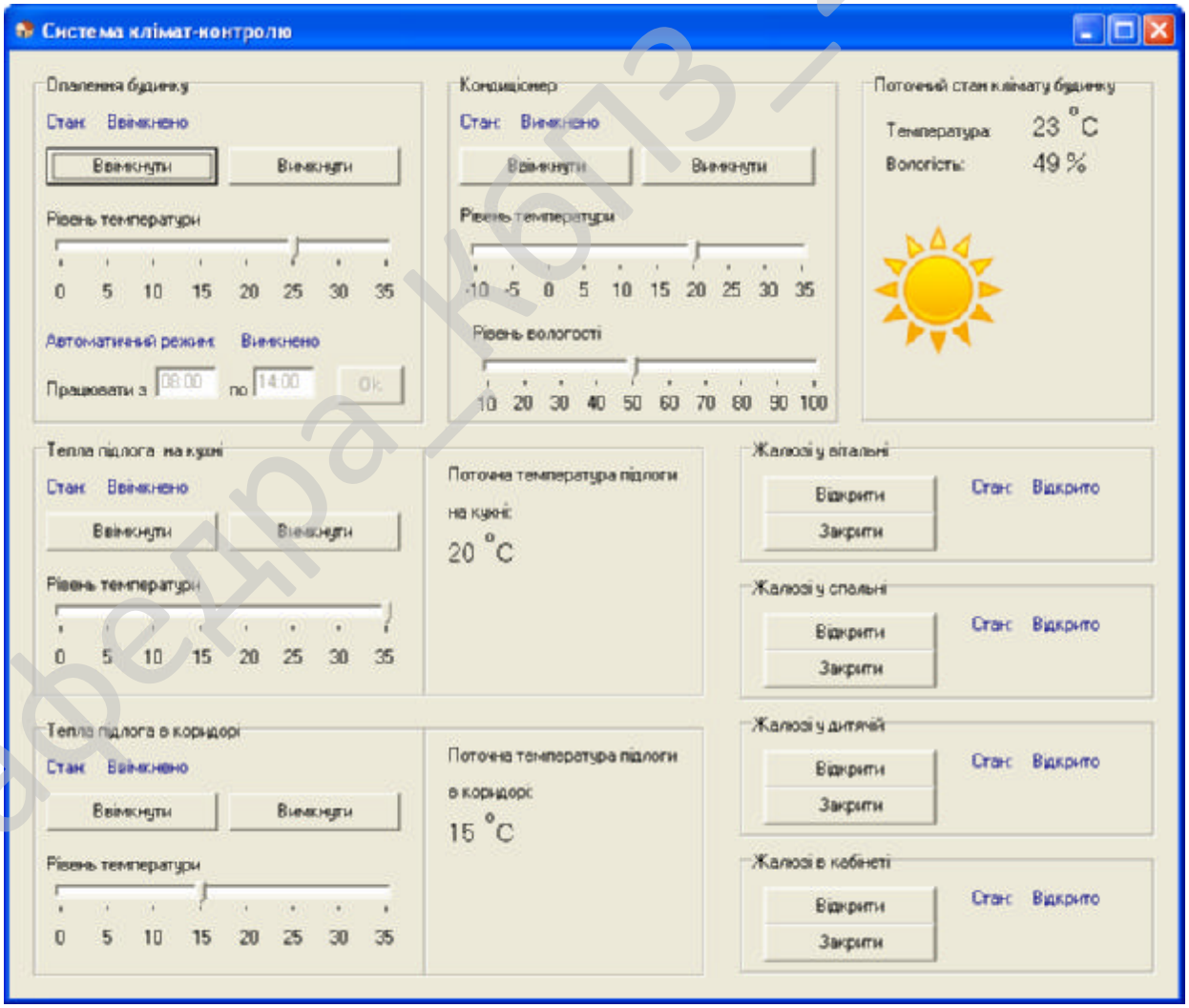


Рисунок 5.6 – Управляння кліматом

Вікно управління телефонним зв'язком зображене на рисунку 5.7.

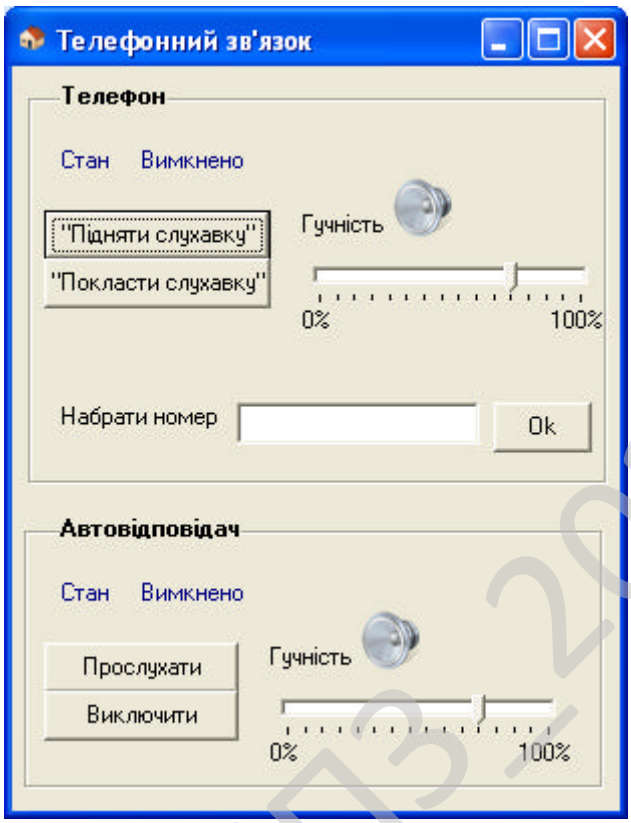


Рисунок 5.7 – Управляння телефонним зв'язком

Вікно управління домашнім кінотеатром зображене на рисунку 5.8.

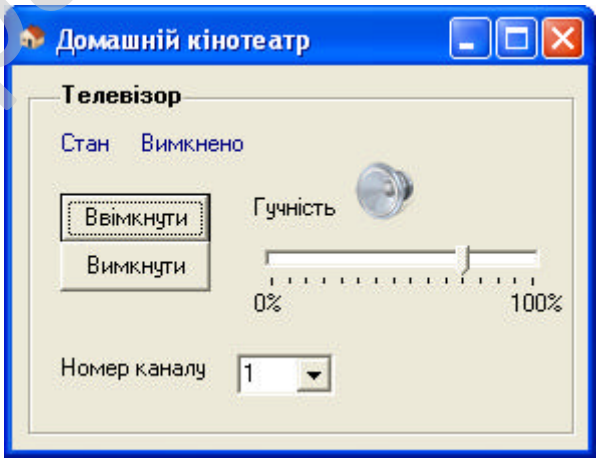


Рисунок 5.8 – Управляння домашнім кінотеатром

Вікно довідки зображене на рисунку 5.9.

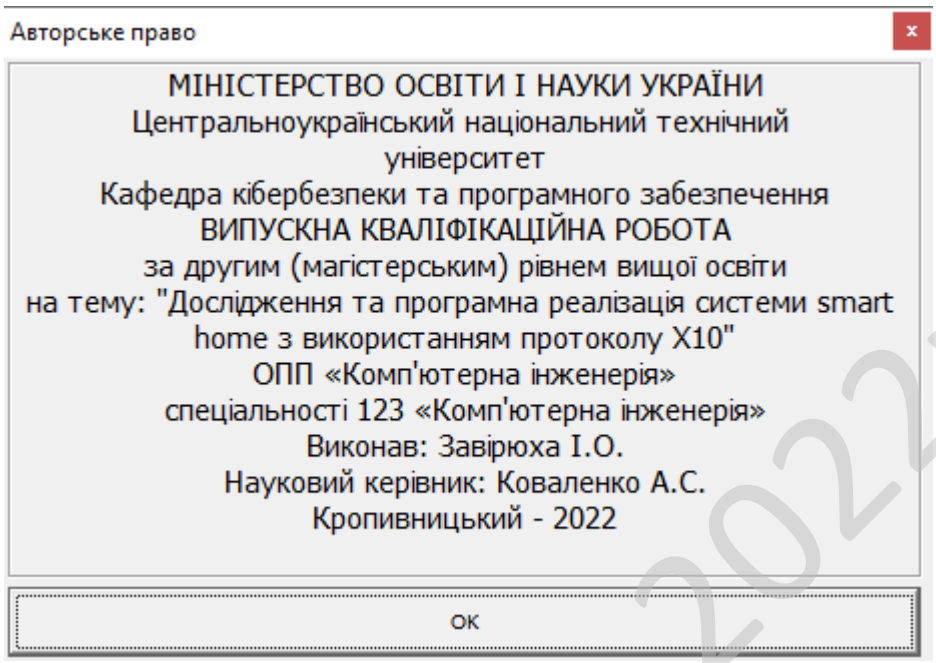


Рисунок 5.9 – Довідка про програму

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи smart home з використанням протоколу X10.

Метою розробки є дослідження та програмна реалізація системи smart home з використанням протоколу X10.

Об'єктом дослідження є процес smart home з використанням протоколу X10.

Предметом дослідження є методи smart home з використанням протоколу X10.

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

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

– Удосконалено метод smart home з використанням протоколу X10.

– Розроблено вітчизняний продукт smart home з використанням протоколу X10, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи smart home з використанням протоколу X10.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	17
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}028} = 6{,}8 \text{ люд-міс.}$$

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

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

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

$$T_{уточн} = 6{,}8 \cdot (0{,}88 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1 \cdot 1{,}1) = 8{,}38 \text{ люд-міс.}$$

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

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

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

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4).

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо S = 90 %

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 8{,}38^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 90 = 177 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	177	Ф 7.1-7.4
Впровадження	17	Д13
Всього	228	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{228 \cdot 1}{48\text{-}5} = 5,3 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	8	240	4
Кабельні господарства ЛОМ на 1 м.п.	2,5	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	45,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{45 \cdot 2}{1{,}2} = 75 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 75/(48 \cdot 8) = 0{,}2 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12135	24270
Продакт-менеджер	0,25	12000	6000
Інженер-програміст	5,3	13000	137800
Інженер-електронщик	0,2	7500	3000
Інженер-системотехнік	0,25	7500	3750
Адміністратор мережі	0,5	7500	7500
Системний програміст	0,25	7500	3750
Дизайнер WEB	0,25	7500	3750
Інженер-верстальник	0,25	7700	3850
Бухгалтер-економіст	0,25	12500	6250
Всього за період розробки	$R_{сп}$ = 8,5	-	$Ф_{роб}$ = 199920

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{199920}{8,5 \cdot 48} = 490 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

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

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

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 9 \cdot 8 \cdot 20000 = 1440000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 144000 грн.

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

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

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

$$І_{нв} = 9 \cdot 3500 = 31500 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми supercomp.kiev.ua за 29.10.22 – джерело http: //supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		12721
Системний блок		7721
Процесор	AMD A8-Series A8-9600 A8 9600 3,1 ГГц 6 Вт, він працює з відеокартою Radeon R (384 шейдерних ядер / 6 обчислювальни ядер) і двоканальним контролером пам'ят DDR4-18600	2000
Системна плата	GIGABYTE B450M S2H V2, 1 x PCI-E 2. x1, 1 x PCI-E 3.0 x16, 4 x USB 3.1 Gen1, 1 VGA, 1 x RJ45, 2 x USB 2.0, 2 x PS/2, 1 DVI-D, 1 x HDMI, 1 x optical SPDIF out, 3 Audio, M.2 2280, 4 x Sata 6.0 Gb/s	1350
Відеокарта	Інтегрована AMD Radeon R7	-
Жорсткий диск	HDD 500 Gb SAMSUNG Barracud HD502HJ (3.5", 500ГБ, 16МБ)	1490
Оперативна пам’ять	DDR4 8GB 2666 MHZ KINGSTO (KVR26N19S8/8)	834
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	416
Корпус	Logicpower 8702 – 550w 12cm	1411
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	220
інше	Клавіатура, мишка	Подарунок

Продовження таблиці 7.5

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Монітор BenQ GL2450HM Black	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

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

Для визначення необхідної кількості капітальних вкладень складем таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	9	12721	11448,9	125937,9
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	147569,4

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
Група 3			
1. Будівлі	1440000	-	-
2. Передавальні пристрої	144000	-	-
Всього по групі	1584000	5	79200
Група 4			
3. Обчислювальна техніка	147570	-	-
Всього по групі	147570	40	59028
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1910976		$А_р$ = 174435,75

Примітка: вартість автомобіля Daewoo Lanos 2007 взята по даним електронного ресурсу https://auto.ria.com/uk/auto_daewoo_lanos_33592444.html та складає 121875 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 490 \cdot 228/17 = 6579 \text{ грн.}$$

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

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

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

$$З_д = 6579 \cdot 10 \cdot 0{,}01 = 658 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(6579+658) = 1592 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

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

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

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

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 5 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 206 \cdot 5 \cdot 1 = 1030 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 17):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 35,6 грн./шт., DVD-R box – 55 грн./шт.

$$З_{М2} = 17 \cdot 55 = 935 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

$$З_{М3} = 500+508+1155 = 2163 \text{ грн.}$$

$$З_М = (1030 + 935 + 2163)/17 = 243 \text{ грн.}$$

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

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

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

$$О_п = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 17 прим.):

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

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

$$A_м = 174436 \cdot 2/(17 \cdot 12) = 1710 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6579
2. Додаткова зарплата виконавців	$З_д$	658
3. Відрахування на соціальні потреби	$С_{оц}$	1592
4. Загальногосподарські витрати	$Г_{осп}$	987
5. Витрати на матеріали	$З_М$	243
6. Освоєння нових операційних систем, мов програмування	$О_п$	987
7. Амортизація основних фондів	$А_м$	1710
8. Повна собівартість програмного забезпечення	$С_п$	12756
9. Плановий прибуток	$П_р$	6378
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	19134
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3826,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	22960,8

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

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

$$C_п = 6579+658+1592+987+243+987+1710 = 12756 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 12756 = 6378 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	22961
Всього капітальних витрат	–	22961

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	173923	57974
2. Витрати на електроенергію	$З_{ел}$	562	389
3. Витрати на амортизацію	$З_{ам}$	0	5740
Всього витрат за рік	$І$	174485	64103

Витрати на обслуговування системи:

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

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 80 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 173923 \text{ грн},$$

до:

$$З_{р\,нов} = 80 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 57974 \text{ грн}.$$

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

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

$$З_{ел\,баз} = 9 \cdot 0{,}15 \cdot 130 \cdot 3{,}2 = 562 \text{ грн}.$$

$$З_{ел\,нов} = 9 \cdot 0{,}15 \cdot 90 \cdot 3{,}2 = 389 \text{ грн}.$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	22961	–	5740,25
Всього відрахувань	-	–	22961	–	5740,25

7.8 Визначення економічної ефективності програмної продукції

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

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

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

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

$$Ев = (19134 - 12756) \cdot 17 - (0,05 \cdot 1584000 + 0,4 \cdot 147570 +$$
$$+0,2 \cdot 121875 + 0,25 \cdot 40531 + 0,1 \cdot 17000) \cdot 2/12 = 79353 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{326976}{(19134 - 12756) \cdot 17 \cdot 12 / 2} = 0,5 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	17
2. Повна собівартість розробленої програми	Грн.	12756
3. Ціна розробленої програми	Грн.	19134
4. Плановий прибуток від реалізації розробленої програми	Грн.	6378
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1910976
7. Загальний прибуток від реалізації програмної продукції	Грн.	108426
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	79353
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	22961
11.Величина економічного ефекту у користувача програмної продукції	Грн.	104642
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

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

$$E_{сп} = (174485\text{-}64103)\text{-}0{,}25 \cdot 22961 = 104642 \text{ грн.}$$

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

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

$$T_{сп} = \frac{22961}{174485 - 64103} = 0{,}2 \text{ } року.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Аналізуючи умови працівників it-сфери, на перший погляд, може здатися, що працівники сфери інформаційних технологій не схильні до ризиків на виробництві, та якщо більш глибоко розглянути умови і специфіку праці фахівців сфері it-індустрії, можна виявити ряд факторів які будуть мати негативний сплив на стан охорони праці, та на самого it-фахівця зокрема. Сюди можна віднести як невідповідність освітлення, так і високий рівень шуму, що негативно позначатимуться як на емоційному так і на фізичному стані фахівця, призводитимуть до зниження ефективності праці та виробничих травм. Також, важливим моментом охорони праці it-фахівця є врахування його психологічних можливостей (швидкість реакції, особливості пам'яті та уваги, емоційний стан, тощо). Для того, щоб забезпечити ефективну роботу it-фахівця, потрібно враховувати та максимально компенсувати такі негативні фактори як: надмірне нервово-емоційне навантаження, довготривалі статичні перевантаження, обмежена рухова активність. Всі ці чинники призводить до різноманітних відхилень у стані здоров’я, зокрема до перевтоми, зниження фізичної та розумової працездатності, неврозів, захворювань серцево-судинної системи тощо. Метою даного розділу є огляд конкретних умов праці спеціаліста у сфері it-індустрії. Завданнями для даного розділу є: аналіз умов праці на робочому місці фахівця it-індустрії, розробка конкретних рекомендацій щодо покрашення умов праці фахівців it-індустрії, огляд пожежної безпеки на it-підприємстві та розрахунок системи загального штучного освітлення виробничого приміщення де працюють ІТ-фахівці.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

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

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

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

У зазначеному приміщенні працюють двоє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-

обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

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

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура,°С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

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

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

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

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

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об'єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов'язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об'єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань

пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об'єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об'єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах [2].

8.5 Розрахункова частина

Система освітлення робочого місця користувача ПК має відповідати наступним вимогам (рис. 8.1).

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 6,4 м, довжина – 8,3 м, висота – 3,2 м.

У зазначеному приміщенні працює 8 людей.

Рисунок 8.1 – Вимоги до системи освітлення робочого місця користувача ПК

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

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

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

S – площа освітлюваного приміщення.

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

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

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

Обчислимо індекс приміщення за формулою:

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

де:

S – площа приміщення, *S* = 53,1 *м*^{2}*;*

h – розрахункова висота підвісу, *h* = 3,2 *м. (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 6,4 м;*

B – довжина приміщення, *B = 8,3 м.*

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=1,1.$$

Знаючи індекс приміщення, за знаходимо *n = 0,46 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8].* Підставимо всі значення у формулу, визначемо світловий потік: *F=57161,7 Лм.*

Для розрахунку дудемо використовувати стельові *світлодіодні панелі* Призма-72 6400К, світловий потік яких $F_л$ = *7200* Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення:

N= 57161,7/ 7200=7,9 шт.

Приймаємо необхідну кількість *світлодіодних світильників* 8 шт.

8.6 Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи smart home з використанням протоколу X10.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів smart home з використанням протоколу X10.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем smart home з використанням протоколу X10.

– Досліджена система smart home з використанням протоколу X10.

– На основі отриманих результатів досліджень створена програмна реалізація системи smart home з використанням протоколу X10.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання smart home з використанням протоколу X10.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Lucifer.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 104642 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Завірюха І.О. Дослідження та програмна реалізація системи smart home з використанням протоколу X10 // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. А. П. Кашкаров. Электронные схемы для "умного дома". Серия: В помощь радиолюбителю – М.: НТ ПРЕСС 2007г., Мягкая обложка, 256 стр. ISBN: 978-5-477-00781-3

3. Гололобов В.Н. "Умный дом" своими руками. NT Press Москва – 2007 – 417

4. И.Федоров, «Сколько этажей у интеллектуального здания?» – "Бизнес: Организация, Стратегия, Системы", №10 1999 г.

5. В. Архипов «Системы для «интеллектуального» здания» – "СтройМаркет", № 45 1999 г.

6. Коваленко А.С. Разработка структуры базы данных интегрированной информационной системы / А.С. Коваленко, А.В. Коваленко // Информационные технологии и защита информации в информационно-коммуникационных системах: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – С. 54-64.

7. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

8. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

9. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко ,

О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

10. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

11. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

12. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

13. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

14. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

15. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

16. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов,

О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

17. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

18. Коваленко А.С. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

19. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

20. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

21. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

22. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку

озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

23. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

24. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

25. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

26. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28-31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

27. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

28. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко //

Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

29. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

30. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

31. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: І всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

32. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (IT & I): II між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

33. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

34. Королюк Н.А. Оценка временных интервалов работы лица, принимающего решение, на автоматизированном командном пункте /

Н.А. Королюк, А.И. Тимочко // Системи обробки інформації. – Х.: ХУПС, 2005. – Вип. 8 (48). – С. 51-54.

35. Костерев В.В. Надёжность технических систем и управление риском: учебн. пособ. / В.В. Костерев. – М.: МИФИ, 2008. – 280 с.

36. Костюков А.В. Підвищення операційної ефективності підприємств на основі моніторингу в реальному часі. / А.В. Костюков, В.М. Костюков. – М.: Машинобудування, 2009. – 192 с.

37. Лазарев А.А. Выбор показателя затрат для анализа сравнительной экономической эффективности техники конечного потребления / А.А. Лазарев, М.В. Бейлин // Сборник научных трудов ХГПУ.– Х.: ХГПУ, 1999. – Вып. 74. – С. 27-29.

38. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 1. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 308 с.

39. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 2. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 208 с.

40. Лапсарь А.П. Метод оценки состояния сложных технических объектов для синтеза быстродействующих прогнозирующих систем / А.П. Лапсарь // Измерительная техника. – 2004. – № 2. – С. 7-10.

41. Линейные задачи оптимизации: Учеб. пособие / С.В. Лутманов. – Пермь: ЛИТЕР-А, 2004. – Ч.1. – Линейное программирование. – 128 с.

42. Литвак Б.Г. Экспертные технологии в управлении. Учебное пособие / Б.Г. Литвак. – М.: Дело, 2014. – 318 с.

43. Локазюк В.М. Надійність, контроль, діагностика і модернізація ПК: Посібн. / В.М. Локазюк, Ю.Г. Савченко. – К.: Видавничий центр «Академія», 2004. – 376 с.

44. Лопатников Л. И. Экономико-математический словарь: Словарь современной экономической науки / Л.И. Лопатников. – М.: Дело, 2003. – 520 с.

45. Манухина С.Ю. Инженерная психология и эргономика: хрестоматия / С.Ю Манухина. – М.: Изд. центр ЕАОИ, 2009. –224 с.

46. Мартыненко М.В. Человекомашинные процедуры поддержки организационно–управленческих решений: учеб. пособие СПбГЭТУ / М.В. Мартыненко, О.И. Шеховцов. – СПб, 2012. – 250 с.

47. Мунипов О.В. Эргономика: человекоориентированное проектирование техники, программных средств и среды: Учебник / О.В. Мунипов, В.П. Зинченко. – М.: Логос, 2001. – 356 с.

48. Надеев А.И. Математическая модель эксплуатационной надежности интеллектуальных датчиков / А.И. Надеев, Р.А. Юсупов, Ю.К. Свечников, Д.Р. Юсупов // Измерительная техника. – М: Стандартинформ, 2004. – № 1. – С. 8-11.

49. Надійність техніки. Аналіз надійності. Основні положення: ДСТУ 2861-94 – [Чинний від 1997–01–01]. – Київ: Держстандарт України, 1995. – 33 с. – (Національний стандарт України).

50. Надійність техніки. Терміни та визначення: ДСТУ 2860-94 – [Чинний від 1996–01–01]. – Київ: Держстандарт України, 1994. – 36 с. – (Національний стандарт України).

51. Нейлор К. Как построить свою экспертную систему / К. Нейлор. – М.: Энегроатомиздат, 2007. – 242 с.

52. Николаева И. П. Экономический словарь / И.П. Николаева. – Проспект, 2015. – 399 с.

53. Онищук А.Г. Радиоприемные устройстваю: Учебн. пособ. – 2-е изд., испр. / А.Г. Онищук, И.И. Забеньков, А.М. Амелин. – Минск: Новое знание, 2007. – 240 с.

54. Осипов В. Базы данных и Delphi. Теория и практика / В. Осипов. – БХВ-Петербург, 2011. – 752 с.

55. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-

007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

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

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

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

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

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

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

62. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

63. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0010.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Завірюха І.О.			Дослідження та програмна реалізація системи smart home з використанням протоколу X10	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Коваленко А.С.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи smart home з використанням протоколу X10.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи smart home з використанням протоколу X10.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи smart home з використанням протоколу X10;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 107 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Коваленко А.С.

Дослідження та програмна реалізація

системи smart home з використанням протоколу X10

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2022 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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendX10Command(0,1,18,1);  //0-код будинку; 1-код пристрою, що
керує ліхтарем; 18-код команди "on"; 1-кількість повторних надсилань команди
ClientSocket->SendLevitonX10(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->SendX10Command(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->SendX10Command(0,2,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,3,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,4,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,5,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,6,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,7,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,8,18,1);
```

```
ClientSocket->SendLevitonX10(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->SendX10Command(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->SendX10Command(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->SendX10Command(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->SendX10Command(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->SendX10Command(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->SendX10Command(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->SendX10Command(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->SendX10Command(0,1,18,1);
ClientSocket->SendLevitonX10(0,1,TrackBar_torch->Position);

corridor->Caption="Ввімкнено";
Image_corridor_on->Visible=true;
Image_corridor_off->Visible=false;
ClientSocket->SendX10Command(0,2,18,1);
ClientSocket->SendLevitonX10(0,2,TrackBar_corridor->Position);

drawing_room->Caption="Ввімкнено";
Image_drawing_room_on->Visible=true;
Image_drawing_room_off->Visible=false;
ClientSocket->SendX10Command(0,3,18,1);
ClientSocket->SendLevitonX10(0,3,TrackBar_drawing_room->Position);

kitchen->Caption="Ввімкнено";
Image_kitchen_on->Visible=true;
Image_kitchen_off->Visible=false;
ClientSocket->SendX10Command(0,4,18,1);
ClientSocket->SendLevitonX10(0,4,TrackBar_kitchen->Position);

cabinet->Caption="Ввімкнено";
Image_cabinet_on->Visible=true;
Image_cabinet_off->Visible=false;
ClientSocket->SendX10Command(0,5,18,1);
ClientSocket->SendLevitonX10(0,5,TrackBar_cabinet->Position);

bedroom->Caption="Ввімкнено";
Image_bedroom_on->Visible=true;
Image_bedroom_off->Visible=false;
ClientSocket->SendX10Command(0,6,18,1);
ClientSocket->SendLevitonX10(0,6,TrackBar_bedroom->Position);

baby_room->Caption="Ввімкнено";
Image_baby_room_on->Visible=true;
Image_baby_room_off->Visible=false;
ClientSocket->SendX10Command(0,7,18,1);
ClientSocket->SendLevitonX10(0,7,TrackBar_baby_room->Position);


bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
ClientSocket->SendX10Command(0,8,18,1);
ClientSocket->SendLevitonX10(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->SendX10Command(0,1,19,1);

corridor->Caption="Вимкнено";
Image_corridor_off->Visible=true;
Image_corridor_on->Visible=false;
ClientSocket->SendX10Command(0,2,19,1);

drawing_room->Caption="Вимкнено";
Image_drawing_room_off->Visible=true;
Image_drawing_room_on->Visible=false;
ClientSocket->SendX10Command(0,3,19,1);

kitchen->Caption="Вимкнено";
Image_kitchen_off->Visible=true;
Image_kitchen_on->Visible=false;
ClientSocket->SendX10Command(0,4,19,1);

cabinet->Caption="Вимкнено";
Image_cabinet_off->Visible=true;
Image_cabinet_on->Visible=false;
ClientSocket->SendX10Command(0,5,19,1);

bedroom->Caption="Вимкнено";
Image_bedroom_off->Visible=true;
Image_bedroom_on->Visible=false;
ClientSocket->SendX10Command(0,6,19,1);

baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
ClientSocket->SendX10Command(0,7,19,1);


bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
ClientSocket->SendX10Command(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->SendX10Command(0,x,18,1);
y=random (6)+2;
ClientSocket->SendX10Command(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->SendLevitonX10(0,1,TrackBar_torch->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення коридору
void __fastcall TForm_light::TrackBar_corridorChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_corridor->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення вітальні
void __fastcall TForm_light::TrackBar_drawing_roomChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_drawing_room->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення кухні
void __fastcall TForm_light::TrackBar_kitchenChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_kitchen->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення кабінету
void __fastcall TForm_light::TrackBar_cabinetChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_cabinet->Position);
}
//---------------------------------------------------------------------------
```

```
//зміна яскравості освітлення спальні
void __fastcall TForm_light::TrackBar_bedroomChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_bedroom->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення дитячої
void __fastcall TForm_light::TrackBar_baby_roomChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_baby_room->Position);
}
//---------------------------------------------------------------------------

//зміна яскравості освітлення ванної
void __fastcall TForm_light::TrackBar_bathroomChange(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,2,TrackBar_bathroom->Position);
}
//---------------------------------------------------------------------------

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

ClientSocket->SendX10StatusQuery();

st=X10State->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=X10State->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=X10State->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=X10State->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=X10State->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=X10State->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=X10State->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=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendX10Command(0,21,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button4Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,21,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button1Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,22,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button2Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,22,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button5Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,23,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button6Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,23,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button7Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,24,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button8Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,24,19,1);
}
//---------------------------------------------------------------------------

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

```
ClientSocket->SendX10Command(0,26,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button10Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,26,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button11Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,27,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button12Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,27,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button13Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,28,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button14Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,28,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button15Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,29,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::Button16Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,29,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar1Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,21,TrackBar1->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar2Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,22,TrackBar2->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar3Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,23,TrackBar3->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_climate::TrackBar4Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,24,TrackBar4->Position);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm_climate::TrackBar5Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(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->SendX10StatusQuery();
st=X10State->GetStateOnOff(0, 21);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=X10State->GetStateOnOff(0, 22);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=X10State->GetStateOnOff(0, 23);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=X10State->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=X10State->GetStateOnOff(0, 26);
if(st="On") Label_5->Caption="Відкрито"; else Label_5->Caption="Закрито";
st=X10State->GetStateOnOff(0, 27);
if(st="On") Label_6->Caption="Відкрито"; else Label_6->Caption="Закрито";
st=X10State->GetStateOnOff(0, 28);
if(st="On") Label_7->Caption="Відкрито"; else Label_7->Caption="Закрито";
st=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendX10Command(0,9,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану холодної води у ванні
void __fastcall TForm_water::Button6Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,10,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану гарячої води у ванні
void __fastcall TForm_water::Button8Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,11,18,1);
}
//---------------------------------------------------------------------------
//ввімкнення системи контролю протікання води на кухні
void __fastcall TForm_water::Button4Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,12,18,1);
}
//---------------------------------------------------------------------------
//відкриття клапану холодної води на кухні
void __fastcall TForm_water::Button10Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,13,18,1);
}
//---------------------------------------------------------------------------

//відкриття клапану гарячої води на кухні
void __fastcall TForm_water::Button12Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,14,18,1);
}
//---------------------------------------------------------------------------

//вимкнення системи контролю протікання води у ванній
void __fastcall TForm_water::Button1Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,9,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану холодної води у ванні
void __fastcall TForm_water::Button5Click(TObject *Sender)
{
```

```
ClientSocket->SendX10Command(0,10,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану гарячої води у ванні
void __fastcall TForm_water::Button7Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,11,19,1);
}
//---------------------------------------------------------------------------

//вимкнення системи контролю протікання води на кухні
void __fastcall TForm_water::Button3Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,12,19,1);
}
//---------------------------------------------------------------------------
//закриття клапану холодної води на кухні
void __fastcall TForm_water::Button9Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,13,19,1);
}
//---------------------------------------------------------------------------

//закриття клапану гарячої води на кухні
void __fastcall TForm_water::Button11Click(TObject *Sender)
{
ClientSocket->SendX10Command(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->SendX10StatusQuery();
st=X10State->GetStateOnOff(0, 9);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=X10State->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=X10State->GetStateOnOff(0, 10);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";
st=X10State->GetStateOnOff(0, 11);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";

st=X10State->GetStateOnOff(0, 13);
if(st="On") Label_5->Caption="Ввімкнено"; else Label_5->Caption="Вимкнено";
st=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendX10Command(0,15,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button1Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,15,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button4Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,16,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_gas::Button3Click(TObject *Sender)
{
ClientSocket->SendX10Command(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->SendX10StatusQuery();
st=X10State->GetStateOnOff(0, 15);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendLevitonX10(0,32,TrackBar1->Position);
}
//---------------------------------------------------------------------------


void __fastcall TForm_tv::onClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,32,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_tv::offClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,32,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_tv::ComboBox1Change(TObject *Sender)
{
ClientSocket->SendX10Command(0,33,23,ComboBox1->Text);
}
//---------------------------------------------------------------------------
//визначення та виведення стану телевізору

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

ClientSocket->SendX10StatusQuery();
st=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendLevitonX10(0,30,TrackBar1->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::TrackBar2Change(TObject *Sender)
{
ClientSocket->SendLevitonX10(0,31,TrackBar2->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::t_onClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,30,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::t_offClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,30,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::a_onClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,31,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::a_offClick(TObject *Sender)
{
ClientSocket->SendX10Command(0,31,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_phone::Button3Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,30,23,Edit1->Text);
}
//---------------------------------------------------------------------------

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

```
ClientSocket->SendX10StatusQuery();
st=X10State->GetStateOnOff(0, 30);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";

ClientSocket->SendX10StatusQuery();
st=X10State->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 "CPUX10Socket"
//---------------------------------------------------------------------------
#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->SendX10Command(0,17,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button20Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,18,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button16Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,19,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button18Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,19,18,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button13Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,15,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button19Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,16,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button15Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,17,19,1);
}
//---------------------------------------------------------------------------

void __fastcall TForm_security::Button17Click(TObject *Sender)
{
ClientSocket->SendX10Command(0,18,19,1);
}
//---------------------------------------------------------------------------

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

ClientSocket->SendX10StatusQuery();
st=X10State->GetStateOnOff(0, 17);
if(st="On") Label_1->Caption="Ввімкнено"; else Label_1->Caption="Вимкнено";
st=X10State->GetStateOnOff(0, 18);
if(st="On") Label_2->Caption="Ввімкнено"; else Label_2->Caption="Вимкнено";

st=X10State->GetStateOnOff(0, 19);
```

```
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=X10State->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 CPUX10Socket.cpp – протокол X10

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

#include <basepch.h>

#pragma hdrstop

#include "CPUX10Socket.h"

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

static inline void ValidCtrCheck(TCPUX10Socket *)
{
    new TCPUX10Socket(NULL);
}
//#include "OcelotStateUnit.cpp"
//------- ------------------
__fastcall TCPUX10Socket::TCPUX10Socket(TComponent* Owner)
    : TClientSocket(Owner)
{
X10StateChangeNum=new TX10StateChangeNum;
OcelotStateChangeNum=new TOcelotStateChangeNum;
X10State=new TX10State;
OcelotState=new TOcelotState;

//Початкова ініціалізація змінних
X10StateChangeNum->Num=0;
OcelotStateChangeNum->Num=0;
FAuth=1;

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

}

__fastcall TCPUX10Socket::~TCPUX10Socket(void)
{
delete X10StateChangeNum;
delete OcelotStateChangeNum;
delete X10State;
delete OcelotState;

}
//------- ------------------
namespace CPUX10Socket
{
    void __fastcall PACKAGE Register()
    {
         TComponentClass classes[1] = {__classid(TCPUX10Socket)};
         RegisterComponents(" CPU-XA", classes, 0);
    }
}
//------- ------------------
void __fastcall TCPUX10Socket::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 TCPUX10Socket::MyOnDisconnect(System::TObject* Sender,
TCustomWinSocket* Socket)
{
SendBuf("d",1);
FAuth=2;
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,FAuth);
}

//------- -------------------
void __fastcall TCPUX10Socket::MyOnError(System::TObject* Sender,
TCustomWinSocket* Socket, TErrorEvent ErrorEvent, int &ErrorCode)
{
if (FOnChangeAuthStatus)
    FOnChangeAuthStatus(this,5);
if (OnMyError)
    OnMyError(Sender,Socket,ErrorEvent,ErrorCode);
}
//------- -------------------
void __fastcall TCPUX10Socket::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(&X10StateChangeNum->Num,&local_pointer[1],2);
                X10State->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 TCPUX10Socket::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 TCPUX10Socket::SendOcelotStatusQuery(void)
{
char buf[3];
buf[0]='q';
memcpy(&buf[1],&OcelotStateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
void __fastcall TCPUX10Socket::Send10StatusQuery(void)
{
char buf[3];
buf[0]='x';
memcpy(&buf[1],&X10StateChangeNum->Num,2);
SendBuf(buf,3);
}
//------- -------------------
unsigned short __fastcall TCPUX10Socket::GetDataFromUnit(unsigned char
UnitNumber)
{
return OcelotState->GetDataFromUnit(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPUX10Socket::GetIO(unsigned char UnitNumber,unsigned
char Point)
{
return OcelotState->GetIO(UnitNumber,Point);
}
//------- -------------------
unsigned char __fastcall TCPUX10Socket::GetUnitVer(unsigned char UnitNumber)
{
return OcelotState->GetUnitVer(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPUX10Socket::GetUnitType(unsigned char UnitNumber)
{
return OcelotState->GetUnitType(UnitNumber);
}
//------- -------------------
unsigned char __fastcall TCPUX10Socket::GetUnitsCount(void)
{
return OcelotState->GetUnitsCount();
}
//------- -------------------
TDateTime __fastcall TCPUX10Socket::GetCPUXADateTime()
{
return OcelotState->GetCPUXADateTime();
}
//------- -------------------
unsigned short __fastcall TCPUX10Socket::GetVariable(unsigned char VarNumber)
{
return OcelotState->GetVariable(VarNumber);
}
//------- -------------------
unsigned short __fastcall TCPUX10Socket::GetTimer(unsigned char TimerNumber)
{
return OcelotState->GetTimer(TimerNumber);
}
//------- -------------------
```

```
TDateTime TCPUX10Socket::GetTimerAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetTimer(VarNumber));
}
//------- -------------------
TDateTime TCPUX10Socket::GetVarAsTime(unsigned char VarNumber)
{
  return UShortToTime(OcelotState->GetVariable(VarNumber));
}
//------- -------------------
TDateTime TCPUX10Socket::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 TCPUX10Socket::SetTimeAsVar(unsigned char VarNumber, TDateTime time)
{
  SetVariable(VarNumber,TimeToUShort(time));
}
//------- -------------------
void TCPUX10Socket::SetTimeAsTimer(unsigned char VarNumber, TDateTime time)
{
  SetTimer(VarNumber,TimeToUShort(time));
}
//------- -------------------
unsigned short TCPUX10Socket::TimeToUShort(TDateTime time)
{
   Word Hour, Min, Sec, MSec;
   DecodeTime(time, Hour, Min, Sec, MSec);
   return Hour*100+Min;
}
//------- -------------------
void __fastcall TCPUX10Socket::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 TCPUX10Socket:: 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 TCPUX10Socket::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 TCPUX10Socket::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 TCPUX10Socket::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 TCPUX10Socket::SendIr(unsigned short Value)
{
char buf[4];
buf[0]='c';
buf[1]=5;
memcpy(&buf[2],&Value,2);
SendBuf(buf,4);
}
//------- ------------------
void __fastcall TCPUX10Socket::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 TCPUX10Socket::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 TCPUX10Socket::GetInternalProtocolVersion(void)
```

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

void __fastcall TX10StateChangeNum::Next(void)
{
if (Num==65534)
    Num=1;
else
    Num++;
};
//------- -------------------
__fastcall TX10StateChangeNum::TX10StateChangeNum(void)
{
Num=1;
};
//------- -------------------
AnsiString __fastcall TX10State::GetStateOnOff(unsigned char house, unsigned
char key)
{
if (map[house][key]==true)
return "On";
else
return " --";
}
//------- -------------------
bool __fastcall TX10State::GetOnOffStatus(unsigned char house, unsigned char
key)
{
if (map[house][key]==true)
return true;
else
return false;
}
//------- -------------------

bool __fastcall TX10State::LoadInfoBuff(char *buff)
{
memcpy(map,buff,256);
return true;
}
//------- -------------------

void _fastcall TCPUX10Socket::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 TCPUX10Socket::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 CPUX10Socket.h - бібліотека для файлу CPUX10Socket.cpp

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

#ifndef CPUX10Socket
#define CPUX10Socket
//------- ------------------
#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 TX10StateChangeNum {
public:           // User declarations
unsigned short Num;
void __fastcall Next(void);
__fastcall TX10StateChangeNum(void);
};

class TX10State {
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 TCPUX10Socket : public TClientSocket
{
private:
    AnsiString FVersion;
    AnsiString FLogin;
    AnsiString FPassword;
    unsigned char FAuth;
    TOcelotStateChangeNum *OcelotStateChangeNum;
    TX10StateChangeNum *X10StateChangeNum;
    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 TCPUX10Socket(TComponent* Owner);
    __fastcall ~TCPUX10Socket(void);
    TX10State *X10State;
    // 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
```