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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Сопілка Б.Ю.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

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

Сопілці Богдану Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Сопілка Б.Ю. Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Метою розробки є дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Об'єктом дослідження є процес мережного керування помешканнями з використанням протоколу Modbus/RTU.

Предметом дослідження є методи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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

Результат роботи – програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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'ютерні науки, Modbus/RTU

ABSTRACT

Sopilka B.Yu. Research and software implementation of a network management system for residences using the Modbus/RTU protocol. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the network management system of residences using the Modbus/RTU protocol.

The purpose of the development is the research and software implementation of the network management system of residences using the Modbus/RTU protocol.

The object of research is the process of network management of residences using the Modbus/RTU protocol.

The subject of research is the methods of network management of residences using the Modbus/RTU 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 network management system for residences using the Modbus/RTU 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 science, Modbus/RTU

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АРМ	–	автоматизоване робоче місце
АСУ	–	автоматизована система управління
ДБЖ	–	джерело безперебійного живлення
ДКС	–	домашня кабельна мережа
ДУ	–	дистанційне управління
ЕОМ	–	електронно-обчислювальна машина
ЕФ	–	екранна форма
ЗТО	–	звукова трансляція й оповіщення
ІБ	–	інтелектуальний будинок
ІЧ	–	інфрачервоний
ОВК	–	управління опаленням, вентиляцією й кондиціюванням
ОДС	–	оперативна диспетчерська система
ОПС	–	охоронно-пожежна сигналізація
ПДУ	–	пульти дистанційного управління
ПЗ	–	програмне забезпечення
ПЛК	–	програмувальні логічні контролери
ПМО	–	програмно-математичного забезпечення
РК	–	рідкокристалічний
СКК	–	система кабельних комунікацій
ТЗ	–	технічне завдання
EIB	–	європейська інсталяційна шина
X10	–	технологія інтелектуального дому

ВСТУП

Актуальність теми. Останнім часом в Україні розвивається будівництво будинків, оснащених системами інтелектуального керування. В Україні, як і в усьому світі, самими популярними об'єктами для впровадження інтелектуальних технологій є комерційна нерухомість (торгові центри, офісні будинки, банки, готелі), державні будинки (вокзали, аеропорти, спортивні й культурні установи), а також об'єкти домашньої автоматизації. У сучасних будинках, насичених інженерним устаткуванням, системи автоматизації й керування виконують функції забезпечення інженерної безпеки експлуатації будинку, інтеграції інженерних систем і, в остаточному підсумку, визначають рівень стійкості функціонування всього об'єкта.

Ідея автоматизації й об'єднання різних систем керування в рамках однієї інтелектуальної системи стимулювала класифікацію об'єктів по двох сегментах:

– автоматизація будинків (Building Automation) – такі об'єкти називають інтелектуальним будинком;

– автоматизація житла (Home Automation) – ці об'єкти звичайно називають системою «Мережне керування помешканнями з використанням протоколу Modbus/RTU».

Автоматизація будинків спрямована, насамперед, на економію ресурсів і зниження експлуатаційних витрат. Автоматизовані системи для житлових будинків мають на увазі створення затишку, комфорту й зручності для його мешканців.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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

– Огляд існуючих систем мережного керування помешканнями з використанням протоколу Modbus/RTU.

– Дослідження системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

– Програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Об'єктом дослідження є процес мережного керування помешканнями з використанням протоколу Modbus/RTU.

Предметом дослідження є методи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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 мережного керування помешканнями з використанням протоколу Modbus/RTU.

– Розроблено вітчизняний продукт мережного керування помешканнями з використанням протоколу Modbus/RTU, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі мережного керування помешканнями з використанням протоколу Modbus/RTU.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації мережного керування помешканнями з використанням протоколу Modbus/RTU. В інтелектуальному будинку операторам доводиться мати справу з досить більшими обсягами цифрових й графічних даних, звукової і відеоінформації. Для прийняття найбільш ефективних рішень, оператор повинен мати можливість швидко класифікувати дані, що надходять, і реагувати на них. Постійно мінливі технології вимагають регулярного одержання нових знань. У випадку розумного будинку всі рішення приймає хазяїн будинку, а система автоматизації повинна надати як можна більше вичерпну інформацію про що відбувається.

Автоматизації й диспетчеризації в будинку підлягають такі інженерні системи:

– димовидалення й підводу повітря;

– водопостачання й водопідготовки;

– обліку споживання електроенергії й тепла;

– кондиціювання;

– дренажу й каналізації;

– керування освітленням;

– приточної і витяжної вентиляції;

– холодопостачання;

– теплового пункту (включаючи ГВС і опалення);

– керування фанкойлами й опалювальними батареями;

– санкціонованого доступу.

При проектуванні й побудові різних типів будинків необхідно вирішити завдання кваліфікованого вибору основних складових систем життєзабезпечення й безпеки об'єктів, а також їхньої інтеграції.

Як відомо, у функції систем безпеки входить контроль зовнішнього периметра, входу й виходу з об'єкта, керування доступом у приміщення в цілому й внутрішніх зонах, охорона виділених зон, а також протипожежні заходи.

Системи життєзабезпечення призначені для створення комфортних умов роботи персоналу, автоматичної підтримки мікроклімату приміщення й організації взаємодії з підсистемами безпеки при виникненні позаштатної ситуації.

Хочеться підкреслити, що інтеграція елементів систем безпеки й життєзабезпечення відбувається через єдиний пульт керування, що дозволяє розглядати їх як інтегровані системи. Це дає можливість досягти оптимального рівня комфорту, прийнятній економічності й адекватному ступеню захищеності об'єктів, які будуються.

Система диспетчеризації й керування

Великий потік інформації від різних систем, необхідність безпомилкового аналізу й видачі єдино вірних керуючих команд – все це диктує необхідність створення керуючого центра будинку. Із цим завданням справляється система диспетчеризації й керування будинком, що призначена для центрального керування інженерним устаткуванням і мікрокліматом у будинку. При цьому досягається не тільки інтегроване й оптимальне керування інженерними системами, але й значна економія споживаної електроенергії.

Інтелектуальне керування базується на сукупності стандартів і протоколів, які поєднують різні пристрої й прилади в єдиний організм. Пропоновані на ринку системи повинні бути відкритими, підтримувати більшість сучасних комунікаційних протоколів і інтегруватися із системами інших виробників. Зараз на ринку присутні такі відкриті мережні протоколи, як LonWorks, BACNet і ін.

Центральним елементом даної системи є комп'ютер з компресором програмного забезпечення, з'єднаний з локальними контролерами в мережу за допомогою відкритого протоколу. Давайте розглянемо принцип роботи системи:

– Нижній рівень. Кожний контролер з датчиками й виконавчими пристроями управляє певною технологічною установкою (наприклад, центральним кондиціонером, холодильною машиною й т.д.). Як правило, нижній рівень являє собою систему автоматики, що може функціонувати автономно.

– Середній рівень. Це система диспетчеризації об'єкта. Всі контролери й центральний комп'ютер диспетчера з'єднані загальною шиною даних, по якій відбувається обмін інформацією. Після того як інформація від датчиків і виконавчих механізмів надходить на відповідні контролери, вона обробляється ними й передається в диспетчерську. Найбільш важлива інформація відображається на моніторі диспетчера.

– Верхній рівень. Це об'єднання робочих місць диспетчерів у локальну мережу. Така система застосовується на більших територіально рознесених об'єктах.

Диспетчер має можливість регулювати всі параметри інтелектуальної системи, попереджати виникнення аварій, вживати необхідних заходів безпеки й, таким чином, активно брати участь у створенні сприятливих умов роботи й комфортного мікроклімату. Програмні й апаратні засоби робочого місця диспетчера забезпечують:

– спостереження за режимними параметрами регульованих процесів у зручній для оператора формі у вигляді графічних мнемосхем установок і планів приміщень;

– побудову графіків зміни параметрів у реальному часі;

– архівацію всіх даних протягом року для наступного аналізу;

– видачу оператором керуючих команд, складання календарних планів роботи встаткування;

– негайне оповіщення оператора про відмови встаткування;

– обмеження доступу операторів, документування всіх дій, вироблених оператором, захищеність від проникнення в систему сторонніх осіб;

– видачу звітів, що підсумують інформацію з роботи систем будинку за певний строк.

1.2 Область застосування

Областю застосування системи є інтелектуальні будинки. Збільшення сегмента інтелектуальних будинків в Україні обумовлено ростом інвестиційної привабливості подібних проектів – і не тільки за рахунок підвищення зручності й безпеки роботи персоналу, але й завдяки істотному зниженню експлуатаційних витрат протягом усього життєвого циклу будинку. На думку експертів ринку, попит буде підсилюватися й далі в міру усвідомлення інвесторами об'єктивної доцільності впровадженні таких систем.

Серед тенденцій ринку інтелектуальних будинків в Україні можна виділити шість основних:

1. Будівництво житлових багатофункціональних комплексів, обладнаних інженерними системами інтелектуального будинку. Сьогодні будівельні компанії замислюються про нові способи залучення клієнтів, у тому числі й за допомогою сучасних систем автоматизації, споконвічно передбачених у житлових будинках. Розвинена інженерна й інформаційна інфраструктура інтелектуальних будинків дозволяє реалізувати якісно новий рівень надання послуг, що істотно підвищує споживчу цінність житлового комплексу.

2. Перехід від закритих протоколів до відкритого з метою забезпечення сумісності встаткування різних виробників і його інтеграції в єдину систему. Слід зазначити, що в системах автоматизації й керування будинками практично всі провідні виробники встаткування автоматизації (а, отже, і компанії-інтегратори, що пропонують рішення на українському ринку) переходять на відкриті протоколи або забезпечують сумісність із ними через шлюзи.

3. Ріст професійного рівня виконавців, що необхідно для реалізації складних проектів. На українському ринку є достатня кількість компаній і фахівців, що нагромадили певний досвід у реалізації проектів інтелектуальних будинків і здатних ефективно рухатися в цьому напрямку. Більше того, українські фахівці успішно демонструють свої передові розробки в даній області на вітчизняних і міжнародних виставках, причому багато які з них носять інноваційний характер.

4. Зростання обсягів і частки вартості інженерних систем і систем автоматизації в загальній вартості будівельних об'єктів. Розвиток цієї тенденції на цей момент привело до якісної зміни місця й ролі систем автоматизації й керування будинками, з одного боку, і концепції взаємного вв'язування інженерного встаткування об'єктів і організаційно-технічних рішень по експлуатації з використанням систем автоматизації й керування будинками з іншої. У той же час системи автоматизації й керування будинками формують базу для створення нових сервісів для користувачів у рамках об'єкта. Це знаходить вираження в підвищенні споживчої привабливості інтелектуальних будинків, що виражається, зокрема, у зниженні страхових ризиків за рахунок збільшення стійкості таких будинків до різних дестабілізуючих факторів і зниженні витрат на експлуатацію. Іншими слои, у підвищенні ефективності інтелектуальних будинків у порівнянні із традиційними рішеннями.

5. Просування технологій інтелектуального керування в регіони. Слід зазначити, що сьогодні вони зосереджений переважно в Києві, Львові, Одесі й ряді великих індустріальних центрів. Однак в останні роки простежується тенденція просування даної концепції в регіони.

6. Ріст попиту на системи інтелектуального керування, підвищення інформованості. Про інтелектуальні будинки в Україні в останні роки говорять всі частіше. Завдяки інформованості про дані системи й підвищення інтересу до них як з боку компаній-інтеграторів, які займаються їхнім проектуванням і установкою, так і з боку замовників, спостерігається збільшення попиту на цю

продукцію. Споживачі найчастіше знають про можливості таких систем, про комфорт, що вони можуть забезпечити, а тому проблем з визначенням завдань, які необхідно вирішити, стає усе менше.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU, 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

Ви можете запитати, що таке інструменти домашньої автоматизації. Подумайте про це, у вас є розумний вентилятор, лампочка та замок у вашому домі. Усі ці розумні пристрої мають власний набір програм. Більше таких пристроїв означає більше програм на вашому телефоні. Якщо ви запитаєте мене, це звучить як багато, щоб впоратися.

Що ж, тут вам на допомогу приходять засоби домашньої автоматизації. Це програмне забезпечення, яке об'єднує всі ці пристрої під одним дахом, щоб ви могли керувати кількома пристроями з одного місця.

Більшість із цих інструментів використовують Wi-Fi для підключення до інструментів, і ви можете керувати ними на відстані кілометрів.

Чому вам варто вибрати інструмент автоматизації з відкритим кодом

Існують дуже чіткі причини, чому вам слід вибрати інструмент автоматизації з відкритим вихідним кодом замість закритого (пропрієтарного програмного забезпечення).

Інструмент із відкритим вихідним кодом має широку перевагу, оскільки він дозволяє налаштувати все програмне забезпечення відповідно до ваших уподобань, оскільки вам надається доступ до вихідного коду.

Це також дає вам інші додаткові переваги, такі як:

– Економічно ефективним.

– Спільне технічне обслуговування.

– Здатність розробляти індивідуальний досвід.

– Можливо, краща безпека.

– Підвищена ефективність.

Home Assistant

Тепер, коли ми знаємо причини, давайте перейдемо до списку, який дозволить вам оптимізувати розумні пристрої у вашому домі.

Домашній помічник, який був представлений у The Verge, Product Hunt та інших ЗМІ, може стати чудовим вибором для інструменту автоматизації з відкритим кодом. Він сумісний з Mac, Android, Linux, Windows, Odroid, Asus Tinkerboard і навіть Raspberry Pi.

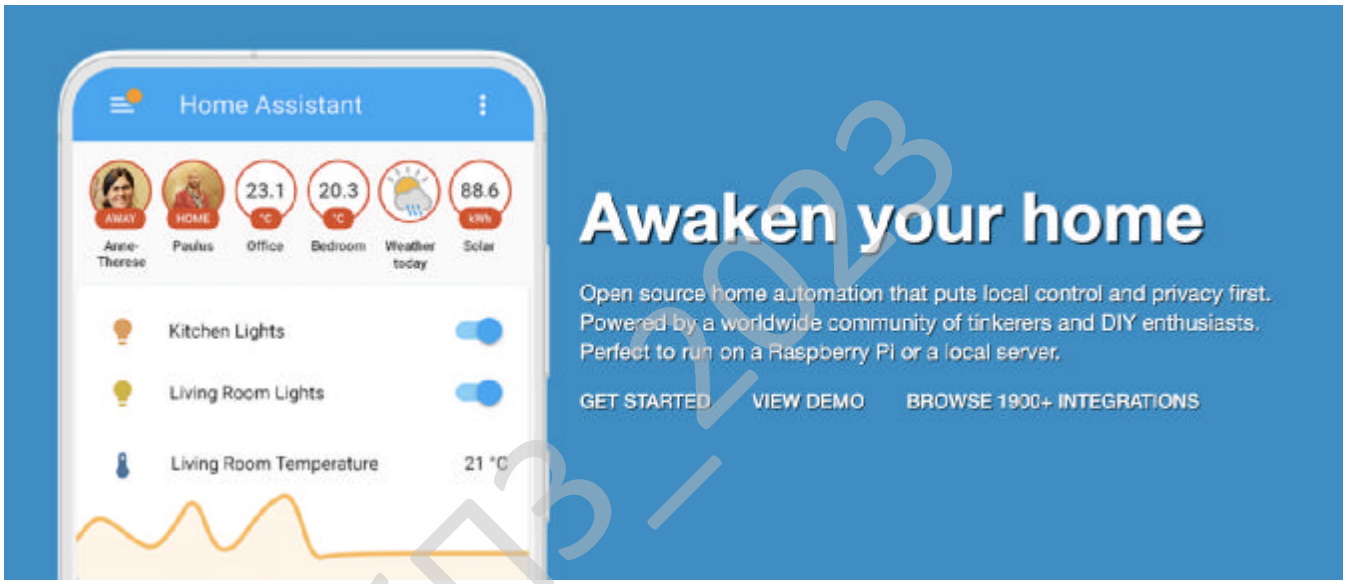

Рисунок 2.1 – Інтерфейс користувача Home Assistant

Інтерфейс інтуїтивно зрозумілий і працює з більш ніж 1000 різними компаніями, що займаються розробкою продуктів для розумного будинку. Ви можете покращити свій досвід за допомогою таких розширень, як блокувальник DNS, NodeRed або Spotify Connect.

Однією з найкращих переваг Home Assistant є те, що він зберігає ваші дані локально і не використовує хмарний сервер. Оскільки в наявності так багато речей, я настійно рекомендую перевірити цей, тому що це може бути все, що ви шукаєте.

Frigate

Якщо ви залишаєтеся параноїком через випадкову систему безпеки камери, вам потрібно взяти Frigate прямо зараз.

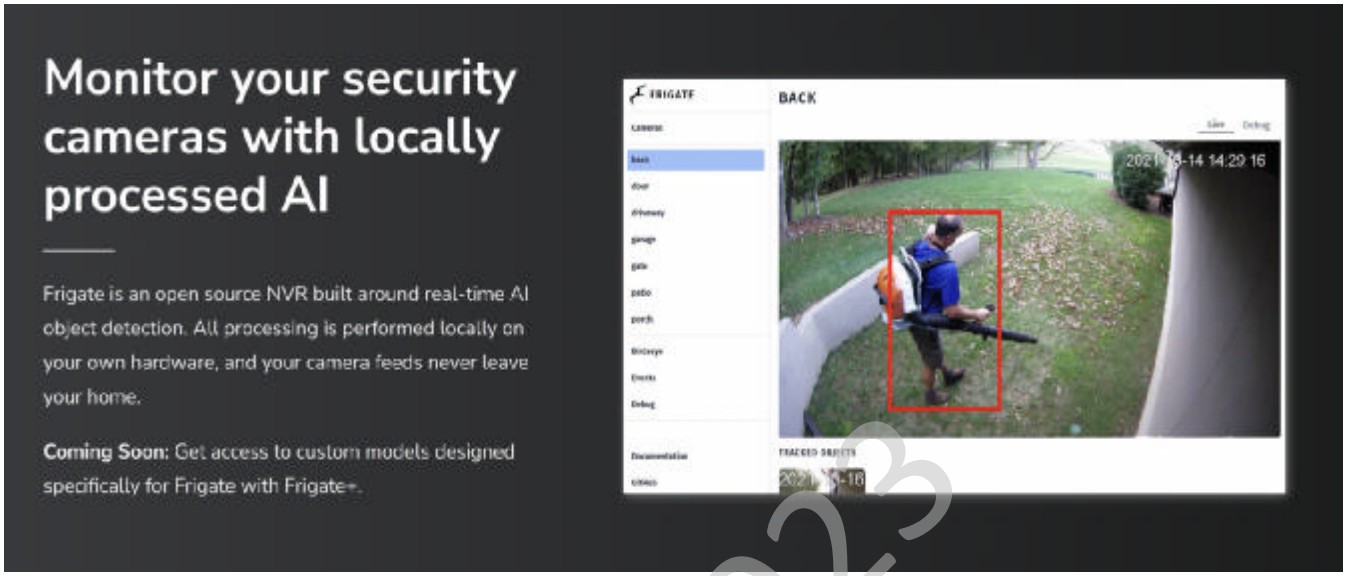

Рисунок 2.2 – Інтерфейс користувача Frigate

Ось деякі з багатьох причин:

– Це допомагає контролювати вашу камеру за допомогою локально обробленого ШІ.

– Ефективно зменшує помилкові спрацьовування.

– Ви можете налаштувати сповіщення відповідно до зон.

– Будьте на шляху за допомогою детальних звітів.

– Добре інтегрується з Home Assistant та іншими розумними пристроями.

У разі будь-яких сумнівів ви можете переглянути динамічний відеозвіт у реальному часі, щоб підтвердити безпеку.

openHAB

Один із найсильніших варіантів для розгляду, openHAB надійний і має величезну базу користувачів з правильних причин. Ви можете легко інтегрувати його з більш ніж 2000 розумними пристроями від найбільших компаній.

Рисунок 2.3 – Інтерфейс користувача openHAB

Їхня програма автоматизації розроблена настільки інтуїтивно зрозумілою, що навіть людина з базовими технічними навичками може легко керувати нею. У будь-який момент часу ви можете скористатися допомогою їх сильної та величезної спільноти, яка завжди активна та ділиться різними порадами та підказками для вашої користі.

Calaos

Calaos – це інструмент, який працює на GPLv3 (проект вільного програмного забезпечення) і дає вам повний контроль над домашньою автоматизацією.

Це допоможе вам створити найкращий досвід для вашого дому:

– Переконайтеся, що світло приглушене, віконниці закриті, а розслаблююча музика готова вітати вас додому після довгого виснажливого дня.

– Дає вам можливість керувати музикою з будь-якого куточка вашого будинку.

– Допомагає створювати та зберігати різні сценарії одним клацанням миші.

Ви можете скористатися наявними інструментами або просто встановити інсталятор Calaos, щоб налаштувати розширені функції. Крім того, він оснащений

чудовою апаратною підтримкою, яка забезпечить вам справді зручне життя.

Domoticz

Для тих, хто новачок у світі технологій, Domoticz стане чудовим вибором. Це зручно для початківців і дає вам повний контроль на кінчиках ваших пальців. Ви можете встановити програмне забезпечення з їхнього офіційного веб-сайту та негайно розпочати роботу.

Ви отримаєте покрокові інструкції щодо того, як завантажити, установити та розпочати роботу прямо на веб-сайті. Існує також розділ для вихідного коду, де ви можете завантажити, змінити та налаштувати його саме так, як вам заманеться.

Ви також можете відвідати постійно активну сторінку спільноти, щоб подивитися, що роблять інші користувачі.

Jeedom

Внесіть інновації у свою домашню автоматизацію за допомогою Jeedom, інструменту, який надає повний доступ із прозорістю та довговічністю.

Деякі з багатьох функцій цього інструменту включають:

- Підтримка кількох протоколів, таких як ZigBee, Modbus, EnOcean тощо.
- Сумісність плагінів.
- Конфіденційна та автономна підтримка.
- Різні варіанти персоналізації.

Він також поставляється з потужними шлюзами домашньої автоматизації у формі двох моделей – Jeedom Altas і Jeedom Altas Pro. У них є чудова команда підтримки, до якої ви можете звернутися, коли вам потрібно вирішити свої запити.

ioBroker

ioBroker – це безкоштовний інструмент, який працює на платформі IoT. Він поставляється з комплексною підтримкою домашньої автоматизації з більш ніж 500 можливими інтеграціями. Ви можете налаштувати або просто використати їхні існуючі графічні інтерфейси, які є приголомшливими.

ioBroker найкраще працює з операційними системами Windows, Mac, OSX, Raspberry Pi та ARM.

Незалежно від того, чи це термостат, освітлення чи жалюзі, будьте впевнені, що це програмне забезпечення сумісне майже з усім, що знаходиться під дахом.

Він надає вам своєчасні звіти для перевірки за допомогою вашої домашньої карти, доступної в інтерфейсі програмного забезпечення, тож ви завжди можете бути в курсі своїх розумних продуктів.

Smarthomatic

Чудова система домашньої автоматизації з відкритим кодом, Smarthomatic проста, але містить усі необхідні функції. Він має зрозумілий і простий у використанні інтерфейс без багатьох химерних елементів, тому вам не буде важко орієнтуватися.

Цей інструмент зашифрований AES, що забезпечує безпечний зв'язок і захист від потенційних зловмисників.

Smarthomatic докладає більше зусиль, щоб повідомити вас, коли рослині потрібна вода. Він також регулює вологість повітря, вимикає конфорки, якщо їх забувають, і допомагає виконувати інші завдання, не потребуючи багато чого.

Вузол-RED

Порівняно з іншими інструментами автоматизації, Node-RED – це програмне забезпечення з відкритим вихідним кодом, яке не потребує програмування, і воно допоможе вам розширити життєвий простір. Він має редактор на основі браузера, який дозволяє легко об'єднувати потоки одним клацанням миші.

Node-RED побудовано на Node.js, який є неблокуючою моделлю та повністю керується подіями. Ви також можете легко імпортувати та експортувати коди, як вам зручно. Крім того, це програмне забезпечення працює як локально, так і в хмарі.

Крім того, ви також можете зберігати коди, шаблони та функції для

повторного використання.

HomeGenie

HomeGenie – це настроюваний, масштабований і портативний інструмент, який працює як магія на Linux, Raspberry AI, Windows, Mac і будь-якому іншому комп’ютері на базі ARMS.

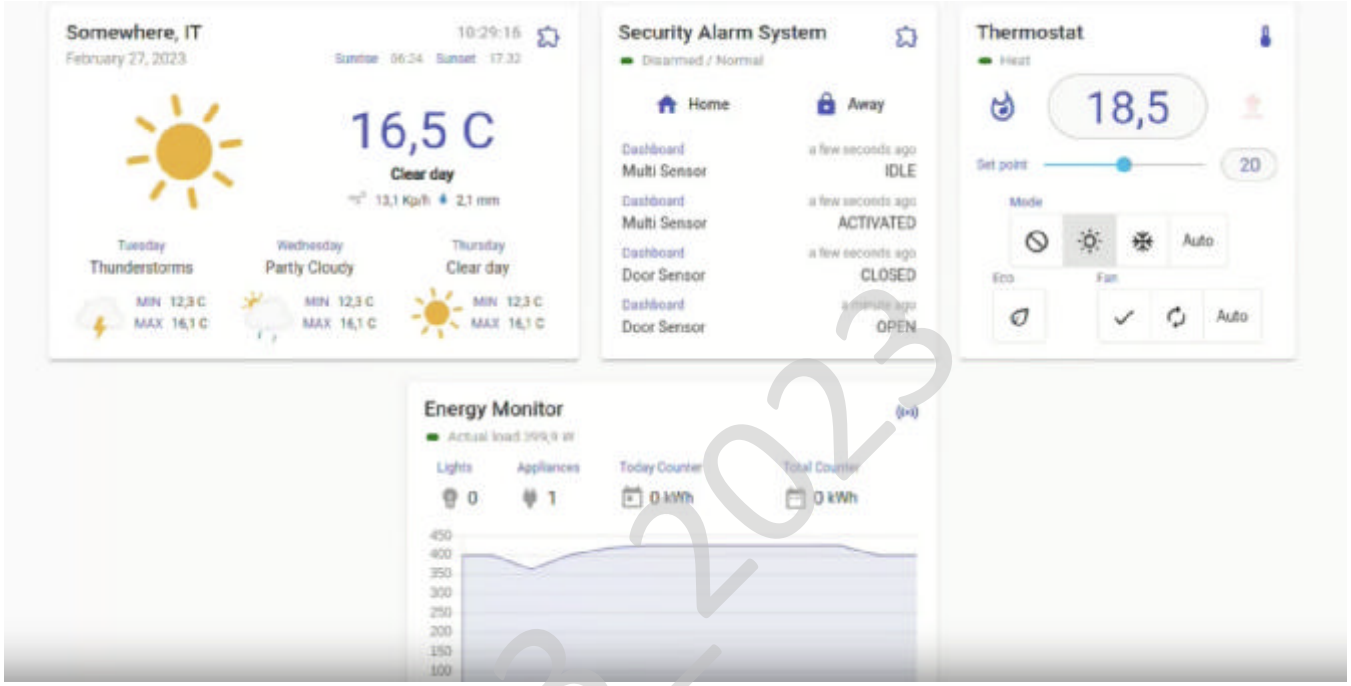

Рисунок 2.4 – Інтерфейс користувача HomeGenie

Він має багаті, але дуже прості функції. Вони включають:

- Обробка даних у режимі реального часу.
- Розширені вирази crontab.
- Сумісність кількох віджетів.
- Прямий доступ до GPIO, I2C, SPI, а також послідовних портів.
- Підтримуються такі основні програми, як C#, JavaScript і Python.

Цей чудовий інструмент домашньої автоматизації можна легко розгорнути протягом кількох хвилин, щоб налаштувати та запустити будь-де.

Homebridge

Управління розумними пристроями з Homebridge плавне, як масло. Він оснащений голосовою інтеграцією, що дозволяє керувати пристроями за допомогою голосових команд через Siri.

Програмне забезпечення також сумісне з вашим Apple Watch, який, на мою думку, є основним джерелом угод.

Інтерфейс зрозумілий і дає вам чіткі вказівки щодо того, що ви хочете виконувати на інформаційній панелі.

Homebridge можна інтегрувати з купою плагінів, і ви можете шукати їх безпосередньо на веб-сайті.

Freedomotic

Freedomotic працює на відкритій основі розробки IoT. Це відкритий вихідний код і надзвичайно зручний для мас, навіть для людей без будь-яких технічних знань.

Це програмне забезпечення, яким користуються та люблять не лише окремі люди, а й компанії. Якщо ви власник бізнесу, це може допомогти вам у таких речах, як розумне середовище роздрібної торгівлі та маркетинг з урахуванням зовнішнього середовища.

Ось деякі з багатьох функцій цього інструменту:

- Постійний унікальний ідентифікатор для легкої ідентифікації речей навколо.
- Підтримка багатьох плагінів для покращення функцій.
- Режим автоматичного виявлення для пошуку об'єктів у реальному часі.
- Відстежує та оновлює будь-які зміни стану в середовищі.
- Багатомовна та багатокористувацька підтримка.
- Дуже безпечний і створений з урахуванням пріоритетів користувачів.

Це дозволяє налаштувати та протестувати програмне забезпечення заздалегідь. Ви можете завантажити найновішу версію та спробувати функції для особистого досвіду.

Безсумнівно, домашня автоматизація – це чудова технологія, якою ми наділені, але керування нею може виявитися складним завданням. З використанням вищезазначених інструментів весь процес стає набагато простішим і менш складним.

Крім того, ви отримуєте повний контроль над ними, все завдяки їх природі з відкритим кодом.

В даний час людство рухається в бік наростання комфорту в наших оселях: сучасна квартира дуже часто крім "арсеналу" побутової техніки та аудіо-відео комплексу містить ще і системи кондиціювання, опалення, освітлення та охорони тощо. Всім цим "електронним організмом" потрібно щодня управляти.

Система «Розумний дім» являє собою роботизовану інженерну систему, здатну за допомогою датчиків управляти функціоналом системи. У найпростішому варіанті реалізується управління інженерними комунікаціями: системою опалення, вентиляції і кондиціонування для забезпечення необхідних параметрів мікроклімату, електрозабезпечення, керування освітленням, системою охорони. Можна з упевненістю сказати, що не існує єдиних рішень, так як функціонал таких систем постійно реорганізується і ускладнюється. Функції «Розумного будинку» можуть бути цілком змінені під вимоги замовника. Незважаючи на дорогі комплексні рішення вони цілком доступні при самостійній збірці і монтажі, ці роботи також можуть бути виконані невеликими компаніями, що спеціалізуються на цифрових рішеннях.

Система мультирум в кімнаті стежить за пересуваннями людини і перерозподіляє звук по приміщенню, функція «Слідкуючий звук»

Концепція «Розумного будинку» або точніше цифрового будинку полягає в розпізнаванні ситуацій, що відбуваються на території моніторингу, в даному випадку будинку або квартири, та організації відповідних реакцій. В якості обов'язкових компонентів розглядаються: система управління доступом в приміщення і спостереження, система управління мікрокліматом, система управління освітленням і водопостачанням, а також система мультирум з

управління розподілом аудіосигналу і т.д. Як правило, такі системи передбачають не тільки звичайне управління, а й управління через он-лайн інтерфейс. Наприклад, вже широко відомі комунальні лічильники з GPS, що дозволяє отримувати дані і управляти своїми платежами дистанційно.

Дорогі системи відрізняються спеціально розробленими девайсами, наприклад, сенсорними екранами, які спрощують керування пристроями з різних місць будинку. Але створити «Мережне керування помешканнями з використанням протоколу Modbus/RTU» можливо, володіючи середнім достатком, це вигідно і зручно.

Існує думка, що встановлювати «Мережне керування помешканнями з використанням протоколу Modbus/RTU» потрібно в будинках із загальною площею, що перевищує 150 м^2, а це класична омана. Комфортне проживання з використанням сучасних технологій доцільно організувати навіть в однокімнатній квартирі. Для цих цілей потрібно придбати спеціальне програмне забезпечення, що керує системою, і девайси, які підключаються в єдину мережу і забезпечують функціонал системи.

«Мережне керування помешканнями з використанням протоколу Modbus/RTU» – пакетні рішення

Бездротові сенсорні панелі Crestron для організації управління «Розумним домом»

За оцінкою фахового журналу «Essential Install» в області систем автоматизації житлових будинків та будівель в справжній момент існує кілька великих виробників, що реалізують у своїй продукції концепцію «Мережне керування помешканнями з використанням протоколу Modbus/RTU», це Crestron, EIB, Domintell, C-Bus та ін

Зазначимо, перш за все, систему Crestron, яка встановлена на яхті Романа Абрамовича. Crestron пропонує велику кількість девайсів для своєї системи і вважається найбільш закінченим рішенням у реалізації складних систем «Мережне керування помешканнями з використанням протоколу Modbus/RTU».

Компанією розроблені стильні сенсорні екрани, які представляють собою бездротові медіацентри, що забезпечують доступ до програмного ядра системи. Crestron пропонує організувати управління своїми системами через мобільний телефон, а також забезпечує колегіальну роботу всіх підсистем.

Окремого коментарю заслуговує технічна сторона системи Crestron. Для реалізації функції управління освітленням розроблений спеціальний цифровий адресний інтерфейс освітлення DALI з функцією енергозбереження. DIN-DALI-2 передбачає управління розподіленими системами освітлення, що може бути використано у великих офісах, торгових центрах. Crestron випускає також бюджетну серію Prodigy. Стартовий пакет, заснований на сенсорних мережах, стандарт ZigBee, запропонований на 700 $. Контролер дозволяє підключити декілька сенсорних пультів, а також 100 бездротових пристроїв. Повний пакет на базі Prodigy буде коштувати 5500 $. Управляється система за допомогою безкоштовного програмного забезпечення Prodigy Composer.

Greenchip NSP – wi-fi – чіп для управління лампочкою для «Розумних будинків»

DOMINTELL – це бельгійська система автоматизації будинку, яка відрізняється більш демократичною вартістю і надійністю рішень з автоматизації будівель. Система підтримує встановлення та управління окремими компонентами системи, це можуть бути сенсорні панелі, вимикачі, датчики та виконавчі пристрої – жалюзі, освітлювальні прилади, електромагнітні клапани, динаміки та вентилятори. Чудово настроюється і легка в монтажі.

Пристрої стандарту X10/S10, що включаються в електромережу, для систем «Мережне керування помешканнями з використанням протоколу Modbus/RTU»

KNX/EIB (європейська інсталяційна шина) – це промислова система, що управляє інженерним устаткуванням будинку на базі стандарту X10/S10. У системі використовуються девайси, які підключаються до електричної мережі і не вимагають додаткової прокладки Ethernet-кабелю. Протоколом EIB можуть бути

реалізовані наступні можливості – включення світла поворотом ключа в замковій щілині, відключення праски за відсутності людини в кімнаті та інше. Датчики управляються за допомогою додаткових протоколів передачі даних. EIB – це досить складна в плані реалізації, але гнучко настроюється система, яка дозволяє створювати індивідуальні рішення з високим рівнем адаптації.

C-Bus – напівпромислова система, яка на відміну від стандарту X10/S10 вимагає прокладки кабелів. Система дуже надійна і забезпечує високу швидкість передачі даних. Даний стандарт підійде для професіоналів, які створюють свої системи автоматизації будинку на базі промислових рішень. Також на промислових шинах працюють системи INSYTE і iNELS (CAN).

Мережеві сервіси для інтелектуального будинку

Компанія Google анонсувала веб-рішення для управління системами «Розумного будинку» за допомогою інтернет і платформи Android @ Home. Задум розробників полягає в тому, щоб надати зручний веб-інтерфейс для управління домашньою електронікою через мобільні телефони для кожного користувача. У проекті бере участь компанія компанія Project Tungsten, яка розробляє акустичні рішення.

Компанія Google також представила новий сервіс для розпізнавання голосу Google Speech API, який може бути з успіхом використаний розробниками, Google пропонує можливість використовувати сервіс у своїх програмах. Хороша якість розпізнавання реалізовано на англійській мові, підключення веб-додатки цілком дає можливість управління «Розумним домом» за допомогою голосових команд.

Біометричні системи для розумного будинку

Біометричні рішення або доступ за відбитками пальців дозволяють організувати систему охорони приміщення на принципово новому рівні конфіденційності. Біометричні системи є невід'ємною частиною «Розумного будинку», хоча на сьогоднішній день і не запропоновані в пакетних рішеннях. Біометричні системи контролю доступу використовуються в системах

відеоспостереження або аутентифікації, ці рішення на порядок перевершують системи на базі пластикових карток RFID.

Якщо розглядати біометричні технології в сукупності, то виділяють дві основні групи в залежності від типу зчитувальних параметрів. Перша група пристроїв працює зі статистичними об'єктами – геометрія руки, відбитки пальців, малюнок сітківки ока, геометрія особи, друга група біометричної ідентифікації працює з динамічними параметрами, до даного типу відноситься ідентифікація за персональним голосовим командам, динаміка підпису і т.д. Біометричні параметри увазі зчитування унікальних характеристик людини, найбільш популярними характеристиками для обробки вважаються райдужна оболонка очей, відбитки пальців, ДНК. Інші характеристики відносяться до змінюваних з часом параметрами, наприклад, почерк або голос.

Найбільш розвиненими технологіями розпізнавання за відбитками пальців вважається продукція компанії BioLink. До бюджетних високотехнологічним рішенням можна віднести продукти китайської компанії ZKSoftware, продукція сертифікована по UL, CE, FCC, ISO.

Сенсорний замок ZKSoftware з OLED-дисплеєм

На даний момент пристроями, які можна використовувати в домашніх умовах для біометричної ідентифікації, є біометричні замки. Цей вид замків представлений кількома виробниками, включаючи компанію ZKSoftware, вартість складає близько 500 $.

«Мережне керування помешканнями з використанням протоколу Modbus/RTU» для квартири

Використовувати нові технології у своїй квартирі доступно для кожного. Перш за все, потрібно визначитися у функціональності системи, саме набір функцій визначає вартість. Причому зовсім не обов'язково реалізовувати всю функціональність відразу. Класичні бюджетні автоматизовані системи складаються з таких реалізацій:

– включення світла з пульта, замість пульта може бути застосований датчик бавовни. Світло може регулюватися по яскравості. З пульта можуть відкриватися штори. Як варіант реалізації світло можна включати по датчику руху з урахуванням розрахунків часу темного часу доби в залежності від сезону. Установка інтелектуального освітлення буде корисна на вулиці, світло спрацьовує на сигнал датчика руху та ІЧ-датчика.

– відеоспостереження та слідкування може бути реалізовано за допомогою звичайного відеореєстратора, відеосистеми і біометричного замку. Біометричні замки дозволяють вводити близько 100 відбитків пальців. Установка веб-камер дозволить спостерігати за квартирою з закритого веб-сайту.

– регулювання мікроклімату доцільно встановлювати в квартирах, що знаходяться на території мегаполісів, це альтернативний варіант установці дорогого кондиціонера з датчиками мікроклімату. Мікроклімат передбачає чистий і рівномірно нагріте повітря без шкідливих домішок, для реалізації цих функції при завданні певних сценаріїв система включає кондиціонери, іонізатори, вентиляцію і відкриває/закриває вікна.

Міні-системи «Мережне керування помешканнями з використанням протоколу Modbus/RTU» передбачають підключення домашнього кінотеатру й імітацію функціональності системи мультирум. Важливим елементом міні-систем є безпечні розетки, які вимикаються при відсутності власників, таке рішення дозволяє захистити побутову техніку від несподіваного стрибка напруги. Найбільш зручно керувати параметрами «Розумного будинку» через он-лайн інтерфейс, який буде містити інформацію про витрату енергоресурсів при підключенні лічильників до системи, контролювати небезпечні стану, включаючи спалаху або протікання, передавати інформацію власнику. Найважливіше, що функціональність «розумного будинку» може бути нарощуваність, причому конфігурація системи вибирається користувачем.

На ринку достатньо широко представлено програмне забезпечення для управління системою «Мережне керування помешканнями з використанням

протоколу Modbus/RTU». При виборі продукту потрібно враховувати, щоб він підтримував протоколи пристроїв Crestron, EIB, Domintell, C-Bus та ін. З декількох пропозицій популярність серед споживачів заслужив програмний продукт QuiiQ Home Automation. Це спеціальне клієнт-серверний додаток, що виконує функції ядра системи «Мережне керування помешканнями з використанням протоколу Modbus/RTU». З інтерфейсу програми можна ефективно керувати зонами і групами. Для управління мультимедійною технікою і створення відео-колекції фото запропонований QuiiQ Movie.

На закінчення підведемо підсумки. Безумовно, представлений огляд дає кожній людині чітке уявлення, яким чином можна покращити свій життєвий простір з використанням цифрових технологій. Всупереч загальній думці, що системи «Мережне керування помешканнями з використанням протоколу Modbus/RTU» – це недоступна для пересічного споживача розкіш, можна з упевненістю стверджувати зворотне. «Мережне керування помешканнями з використанням протоколу Modbus/RTU» можна і потрібно встановлювати практично в будь-якій квартирі, причому ця система допомагає не тільки поліпшити рівень комфорту, а й убезпечити себе від неприємностей, пов'язаних із загорянням, затопленням, перевантаженням мережі і виходом з ладу приладів, це можливість забезпечити абсолютно інший рівень безпеки ваших дітей. На сьогоднішній день системи «Мережне керування помешканнями з використанням протоколу Modbus/RTU» – це доступні для кожного рішення для управління життєвим простором.

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 додатку 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 мережного керування помешканнями з

використанням протоколу Modbus/RTU.

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

Забезпечення ефективної роботи локальної мережі «Розумний дім» вимагає вирішення цілого комплексу завдань:

1. Гетерогенність системи: суміщення декількох технологій передачі даних (протоколів).

Отже, система повинна підтримувати декілька технологій передачі даних, тим самим стаючи більш універсальною.

2. Мультидистанційність: управління системою та отримання інформації про її діяльність за допомогою різних каналів передачі даних (SMS сервіс, Інтернет, ПК).

Необхідність отримання інформації про стан будинку, яку надає локальна система «Розумний дім» може бути необхідна в будь-який момент, так само як і необхідність управління системою. Тому актуальним є не лише управління системою з домашнього ПК, що розташований у даній будівлі, а й за допомогою Інтернет чи мобільного зв’язку.

3. Ієрархічність прав користування системою: система розробляється як багатокористувацька, тож доцільним є функція розподілення прав/ролей для користувачів.

Доступ до системи можуть мати декілька користувачів. Це доцільно і для офісних будівель (можуть бути наявні ролі охоронця, прибиральниці, робітника тощо) і для житлових квартир (батьки, дитина). Різні користувачі, що мають доступ до системи можуть мати різні ролі та наділені відповідними праи.

Отже, метою розроблюваної системи є підвищення ефективності та зручності дистанційного та локального управління системою автоматичного

контролю параметрів мережі «Розумний дім»,розподілення прав/ролей багатокористувацької системи.

Загальна схема роботи розроблюваної системи така:

– центральний мікроконтроллер (або комп'ютер) приймає сигнали від командних пристроїв;

– потім передає ці сигнали виконавчим модулів і систем в будинку;

– виконавчі модулі та системи отримують команди по електромережі, по інфрачервоному або радіоканалу;

– включають або вимикають відповідні пристрої: освітлення, систему охорони, кондиціювання повітря, опалення, подачу води тощо.

Для забезпечення контролю людиною дій системи «Розумний дім» існують такі засоби управління як сенсорні панелі, вимикачі і кнопкові панелі, пульти дистанційного управління, персональний комп’ютер (ПК) тощо. В результаті аналізу існуючих засобів управління з ціллю розвитку та вдосконалення їх щодо розв’язання задач, які були вказані, обрано ПК.

В ході розробки моделі програмного забезпечення системи визначено основні вимоги до системи:

– підтримка декількох каналів передачі даних від центрального мікроконтролера (GSM, USB, COM, TCP/IP);

– підтримка протоколів передачі даних та команд користувача (Internet, SMTP, POP3, GSM(SMS), TCP/IP);

– авторизація та автентифікація користувачів;

– можливість одночасної роботи з однією системою багатьох користувачів з динамічними рівнями доступу (за допомогою різних протоколів);

– можливість швидкого налагодження системи під окрему систему датчиків та керуючих пристроїв, можливість модифікації схеми.

Проблема одночасної підтримки різних каналів даних вирішена за допомогою використання паралельних потоків. Це значно покращує швидкодію отримання даних, адже програма проводить не послідовне опитування каналів

передачі даних на наявність нових повідомлень від центрального контролера, або окремого датчика, а паралельну перевірку каналів. Звичайно, використовувати даний метод краще на ПК на основі багатоядерних процесорів, щоб повністю оцінити всі переваги швидкодії паралельних потоків.

Для вирішення проблеми одночасної роботи з протоколами передачі даних (команд користувача) було використано аналогічний метод. Користувач може завчасно налаштувати ті протоколи, які він буде використовувати та система переходить у стан очікування у паралельних потоках на команду користувача для подальшої їх обробки.

Ієрархічність прав користування системою забезпеченно окремим модулем, за допомогою якого виконується аутентифікація та авторизація користувачів системи. У розроблюваній системі виділено 3 основні групи користувачів: адміністратор (фахівець, що здійснює налагоджування системи), головний користувач (адміністратор серед користувачів) та користувачі з динамічним рівнем доступу.

3.2 Розробка структурної схеми

Система "Мережне керування помешканнями з використанням протоколу Modbus/RTU" – це комплекс інтелектуальної автоматики, який керує абсолютно всіма інженерними системами сучасної будівлі, будь то квартира, будинок або офіс. Основні завдання системи "Мережне керування помешканнями з використанням протоколу Modbus/RTU" – це комфорт і безпека. Система управління "Мережне керування помешканнями з використанням протоколу Modbus/RTU" дозволяє централізовано встановлювати – освітлення, температуру, вологість, доступ і безпеку.

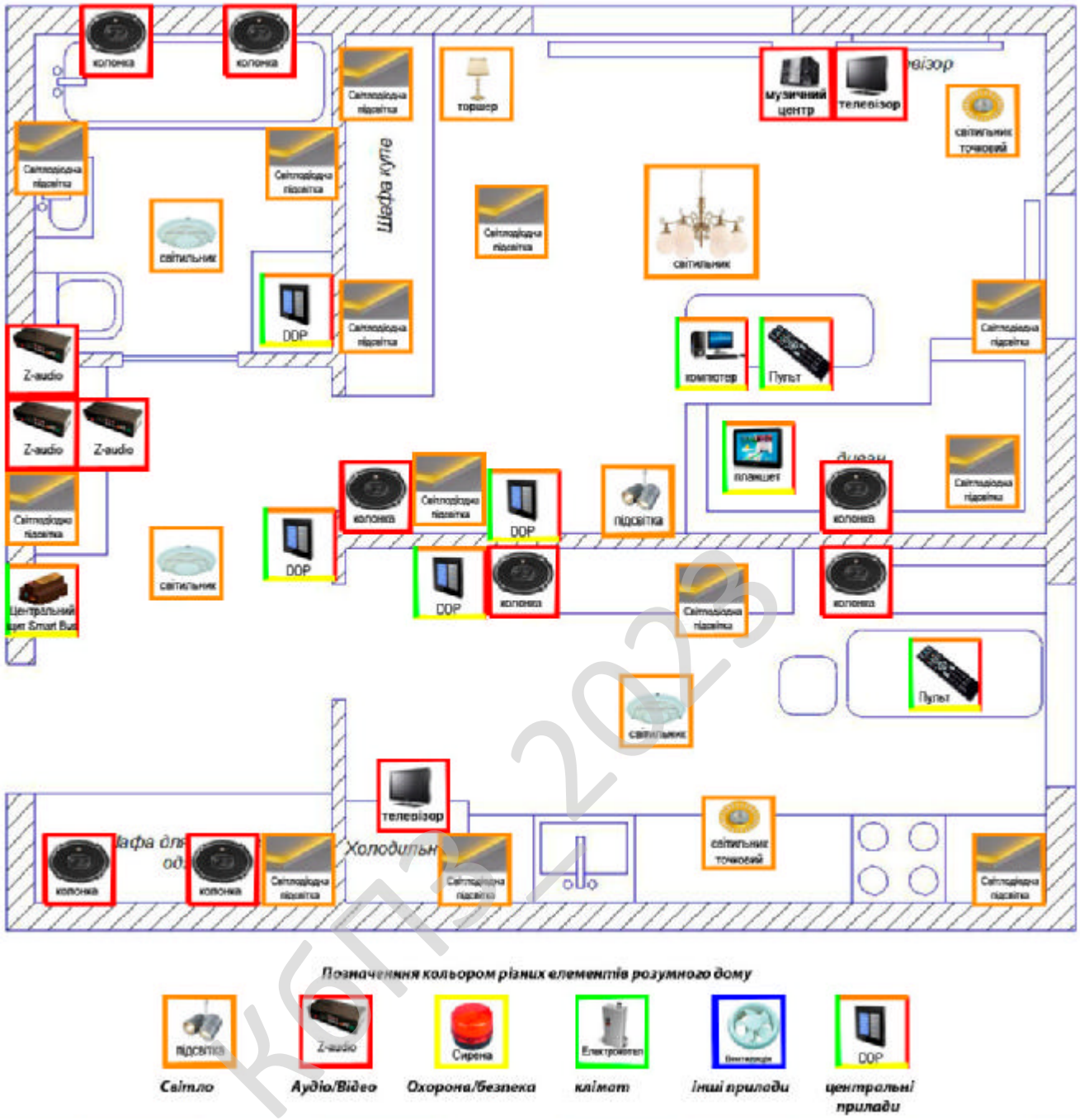

Рисунок 3.1 – Структура схема системи

Автоматизація «розумного дому» є одним із найкращих варіантів використання технології IoT, оскільки ринок B2C для пристроїв IoT стрімко зріс за останні роки. Цей прискорений інтерес до рішень для розумного дому частково викликаний тим, що через COVID-19 люди почали проводити більше часу вдома та хочуть зробити свої домівки затишнішими та зручнішими.

Розвиток автоматизації розумного будинку: важливість надійних технологій IoT

Наразі клієнти B2C обережно вибирають свого індивідуального партнера з домашньої автоматизації через часті випадки, коли компанії з Інтернету речей раптово припиняють свою діяльність, як це сталося з виробником Інтернету речей Insteon . Керівникам не вдалося продати компанію, і в квітні 2022 року Insteon покинула ринок, навіть не попередивши своїх клієнтів. Через це багато людей мали пристрої IoT, які більше не підтримувалися.

Технології IoT зазвичай включають апаратні та програмні компоненти. У цьому розділі ми зосередимося на розробці додатків домашньої автоматизації для ефективного керування пристроями Інтернету речей. Лише за допомогою правильного поєднання апаратного та програмного забезпечення IoT ви зможете надавати виняткові послуги своїм клієнтам. Розробка безпечних і безперебійно функціонуючих технологій IoT може допомогти вам легше знаходити надійних інвесторів і залучати нових клієнтів.

Але починаючи розробку рішення для розумного будинку, ви можете зіткнутися з багатьма проблемами. Давайте обговоримо, як вирішити хоча б деякі з них.

Як подолати загальні виклики впровадження технологій розумного будинку

Незважаючи на зростаючий ринок B2C IoT, багато людей все ще не готові прийняти системи домашньої автоматизації. У цьому розділі ми обговорюємо загальні проблеми, які заважають користувачам використовувати технології Інтернету речей і вимагають додаткових зусиль і інвестицій від постачальників послуг Інтернету речей.

Високі витрати на впровадження. Коли мова заходить про IoT, це завжди стосується вартості. Коли ми думаємо про рішення для розумного будинку, ми зазвичай уявляємо собі великий футуристичний будинок. Багато людей вважають, що розумні будинки – це ідея майбутнього і що лише багаті можуть

дозволити собі технології розумного будинку. Проте ринок IoT розвивався роками, і більшість пристроїв IoT для домашнього використання не коштують цілого стану. Ключовим моментом є розробка простих у навігації та безперебійного функціонування інтелектуальних домашніх систем, щоб виправдати їх вартість для ваших клієнтів. Люди охоче інвестують у високоякісні пристрої IoT, які можуть полегшити їм життя.

Незрозуміле значення. Встановлення пристроїв IoT може здатися радше розвагою, ніж необхідністю. Є пристрої IoT, які використовуються лише для розваг, але є також досить зручні пристрої, які можуть забезпечити безпеку вашого дому, наприклад, коли ви відсутні. Ваше завдання як постачальника послуг IoT полягає в тому, щоб ви надавали цінність своїм клієнтам. Тому під час програмування «розумного будинку» дуже важливо збагатити своє рішення широким і корисним набором функціональних можливостей і постійно вивчати відгуки клієнтів.

Відсутність взаємодії. Незалежно від того, скільки пристроїв IoT у вас є, якщо вони не спілкуються один з одним і ви не можете керувати ними з центрального концентратора, вони не будуть корисними. Важливо надати сучасним клієнтам централізовану платформу управління IoT, яка може служити центром для всіх підключених побутових пристроїв і створювати відчуття інтегрованої та повноцінної системи розумного дому.

Питання безпеки та конфіденційності. Пристрої та рішення IoT часто можуть бути скомпрометовані, якщо вони не захищені належним чином. Користувачі обережно довіряють свої будинки партнерам IoT, які не можуть довести безпеку своїх рішень. Далі в цьому розділі ми обговоримо, як забезпечити безпеку програмного забезпечення IoT. В першу чергу йдеться про налаштування безпечного життєвого циклу розробки програмного забезпечення (S-SDLC) під час розробки програм для iOS або Android для домашньої автоматизації. Також важливо, щоб рішення Інтернету речей відповідали міжнародним і місцевим актам і нормам із захисту даних, як-от GDPR у Європі.

Різні групи користувачів можуть мати різні причини відмови від впровадження технологій IoT. Вивчіть свою цільову аудиторію та вирішуйте її критичні виклики.

Розглянувши проблеми впровадження розумного дому, тепер ми можемо обговорити випадки використання та функціональні можливості програмних рішень для розумного дому.

Розробка додатків для розумного дому: варіанти використання та функціональні можливості

Функціональні можливості, які ви можете включити у своє програмне рішення IoT, залежать від типу та кількості підтримуваних пристроїв IoT. Незалежно від того, чи є ви виробником пристроїв Інтернету речей чи постачальником програмного забезпечення для Інтернету речей, вам потрібно буде створити програмне забезпечення, яке зможе задовольнити більшість потреб ваших кінцевих клієнтів. Ви можете створити розумний домашній продукт, який може стати системою управління будинком для великого набору пристроїв IoT.

Безпека

Безпека часто є основною причиною, чому клієнти вибирають пристрої IoT. Спостереження за вашим будинком на відстані є однією з переваг рішень IoT. Наприклад, IoT-компанія Netatmo надає користувачам розумні камери спостереження та дверні дзвінки. Netatmo Smart Video Doorbell дозволяє користувачам бачити своїх відвідувачів і розмовляти з ними віддалено.

Netatmo також дозволяє клієнтам керувати всіма своїми пристроями безпеки за допомогою єдиного мобільного додатку, а також отримувати сповіщення в режимі реального часу у разі будь-яких підозрілих дій у будинку або поблизу нього. Віддалений домашній моніторинг варто включити в будь-яке програмне рішення IoT.

Автоматизація домашніх справ

Ще одним популярним варіантом використання пристроїв IoT є автоматизація домашніх справ, що може заощадити багато часу. Існує багато

побутової техніки і навіть цілі системи розумного дому, які можуть виконувати домашні справи.

Наприклад, духовка June – це розумна духовка з кількома функціями, якими можна дистанційно керувати з мобільного додатку. Користувачі можуть дистанційно розігрівати свої страви або дивитися відео в прямому ефірі, як вони готуються. Духова шафа має камеру з механізмом розпізнавання їжі, тому, коли їжу поміщають у духовку June, вона визначає, що готується, і пропонує найкращий режим для приготування. Крім того, програма надсилає сповіщення, коли їжа готова.

Іншим прикладом є програма Bosch Home Connect, яка дозволяє клієнтам контролювати всі свої розумні побутові прилади Bosch. Насправді інші виробники, зокрема Siemens, Gaggenau та NEFF, мають подібні програми. Додаток Bosch особливо корисний для керування цілим набором кухонної техніки, щоб спростити процеси приготування та прибирання. Додаток також є чудовим помічником, оскільки пропонує широкий вибір статей та рецептів для спрощення домашніх справ.

Контроль клімату та енергоспоживання

Контроль температури, вологості та якості повітря вдома також вважається важливими функціями домашньої автоматизації. Такі компанії, як Nest і Ecobee, пропонують розумні термостати, які дозволяють регулювати температуру вдома. Ecobee також має функцію eco+, за допомогою якої клієнти можуть попередньо нагрівати та охолоджувати свої домівки перед приходом, підвищуючи енергоефективність. Перегляньте також наш нещодавній проект stromee, створене нами програмне забезпечення для ефективного моніторингу споживання енергії, яке дозволяє користувачам стати більш екологічними, а також заощадити гроші на рахунках за електроенергію.

Пристрої Ecobee не мають власного програмного забезпечення, а підключаються до звичайних рішень, таких як Amazon Alexa, Apple HomeKit і Google Assistant, щоб забезпечити домашню автоматизацію iOS і Android. Однак

використання цих рішень домашньої автоматизації призведе до меншої гнучкості для користувачів і може не дозволити їм отримати доступ до повного потенціалу вашого пристрою. Наприклад, користувачі можуть не мати змоги переглядати історичні графіки температури вдома за певний час.

Розваги

Використання пристроїв IoT для розваг важливо для багатьох клієнтів, особливо для зайнятих сімей з дітьми. Програмні рішення, такі як Roomie Remote, допомагають клієнтам контролювати свої аудіо- та відеосистеми. Використовуючи Roomie Remote, користувачі можуть переглядати аудіо, відео та інші медіа на своєму Apple TV за допомогою жестів або голосу. Крім розважальних функцій, Roomie Remote допомагає користувачам контролювати інші пристрої, такі як термостати, камери спостереження та освітлення.

У наступних розділах ми обговоримо, як створити масштабоване, безпечне та привабливе програмне забезпечення для домашньої автоматизації.

Як побудувати систему розумного будинку

Існує багато виробників пристроїв IoT. Причому у користувачів часто є кілька пристроїв від різних виробників. Їм може бути незручно використовувати окремий мобільний додаток або веб-платформу для кожного пристрою. Набагато простіше, коли всі пристрої IoT підключені, спілкуються один з одним за допомогою єдиного протоколу зв’язку та керуються ними через централізовану платформу IoT.

У попередній статті ми розглянули основні аспекти віддаленого керування Інтернетом речей для мереж великих пристроїв. Ви можете прочитати цю статтю, якщо плануєте надавати свої послуги IoT не лише клієнтам B2C, але й великим підприємствам.

У цьому розділі ми обговорюємо три аспекти розробки системи управління розумним будинком:

– Налаштування хмарного середовища для збору та зберігання даних IoT

– Забезпечення безпеки програмного забезпечення IoT відповідно до галузевих стандартів

– Увімкнення аналізу та візуалізації даних IoT

Хмарна обробка та зберігання даних IoT

Забезпечення належного агрегування, обробки та зберігання даних IoT є важливим елементом у створенні систем управління IoT. Оскільки дані IoT є неструктурованими та часто генеруються в реальному часі, необхідно створити відповідне хмарне середовище. Кілька сервісів AWS, як-от AWS IoT Core та AWS IoT Device Management, дозволяють підключати стільки пристроїв IoT до хмари, скільки необхідно. Ви також можете розглянути нашу статтю про обробку даних у реальному часі, щоб отримати більше інформації про керування даними в реальному часі.

Найбільш підходящим сервісом для зберігання даних IoT є відро Amazon S3 або озеро даних. Озера даних можуть зберігати величезні обсяги неструктурованих даних. У нашому детальному посібнику зі сховищ даних ми обговорюємо, чим озеро даних відрізняється від інших систем зберігання даних, і розповідаємо про його переваги для підприємств.

Щоб навести вам приклад успішного запуску системи IoT у хмарі, ми обрали виробника електроніки Belkin, який почав випускати набір пристроїв для автоматизації розумного будинку. Оскільки кількість розумних домашніх пристроїв і клієнтів зростала, керівництво Belkin зрозуміло, що локальна архітектура компанії IoT не може впоратися з навантаженням, тому вони вирішили перейти на хмарну архітектуру. Результати цього рішення були вражаючими:

Життєвий цикл розробки програмного забезпечення (SDLC) був скорочений більш ніж на 40 відсотків, з 12 місяців до 6,5 місяців.

Компанія заощадила від 30 до 40 відсотків на операційних витратах

Для RAKwireless, виробника пристроїв IoT, ми створили масштабоване рішення IoT за допомогою сервісів AWS IoT Core та AWS Lambda. Завдяки

нашому рішенню RAKwireless вдалося скоротити час, витрачений на налаштування та обслуговування мережі IoT, і тепер вона готова надавати свої послуги набагато більшій кількості підприємств.

Хмарні обчислення можуть бути правильним вибором для розробки вашої платформи управління IoT, якщо ви очікуєте, що ваша компанія буде масштабуватися. Як ми бачили з Belkin, хмарні рішення також скорочують тривалість SDLC і допомагають оптимізувати ваші інвестиції в розробку програмного забезпечення IoT. Насправді серед постачальників послуг Інтернету речей зростає тенденція переносити свою інфраструктуру в хмару. Отже, якщо ви тільки виходите на ринок Інтернету речей, варто створити надійну хмарну інфраструктуру з самого початку, щоб уникнути майбутніх труднощів із хмарною міграцією.

Як зробити програму домашньої автоматизації максимально безпечною

У цьому розділі ми наголошували на тому, що вам слід звернути увагу на розробку максимально безпечної системи розумного дому. Нікому не буде приємно дізнатися, що його камеру спостереження зламали, і тепер будь-хто може проникнути в їхній будинок непоміченим. Щоб випускати безпечні продукти IoT, важливо дотримуватися принципів безпеки на всіх етапах SDLC.

Крім того, може бути корисним дотримуватися певного набору галузевих стандартів, наприклад, архітектури безпеки платформи (PSA).

Відповідно до PSA розробник розумного дому має виконати чотири важливі кроки для розробки безпечного програмного забезпечення IoT.

Аналізуйте. Під час цього кроку команда розробників програмного забезпечення повинна скласти список вимог безпеки за допомогою методів моделювання загроз і виявлення вразливостей.

Архітектор. Цей крок вимагає розробки архітектури безпеки, яка відповідає вимогам PSA та десяти цілям безпеки PSA .

Реалізувати. Наступним кроком є створення програмного забезпечення з архітектурою безпеки та забезпечення безпеки апаратного забезпечення. Також важливо встановити безпечний зв'язок між пристроями IoT і програмними рішеннями.

Сертифікувати. Отримання сертифікату PSA в кінцевому підсумку пов'язане з безпекою технологій IoT. Захист вашої мережі IoT відповідно до стандартів PSA пропонує не лише додатковий захист, але й слугує перевагою продажу, доводячи вашим клієнтам, що ви дбаєте про якість своїх послуг.

Рішення IoT складаються з апаратних і програмних компонентів, і засоби контролю безпеки для них відрізняються. Насправді не існує універсального підходу до безпеки для систем Інтернету речей, тому необхідно налаштувати елементи керування безпекою для кожного апаратного та програмного продукту.

Засоби безпеки для апаратного забезпечення IoT

Безпечне завантаження. Це процес перевірки мікропрограми пристрою IoT за допомогою криптографічних хеш-алгоритмів. Щоб забезпечити Secure Boot, пристрій запрограмовано на ключі безпеки та підписи.

Корінь довіри. Для середовища Secure Boot також потрібен Root of Trust, набір ідентифікаційних і криптографічних ключів, вбудованих в обладнання IoT. Корінь довіри зазвичай вважається серцем пристрою IoT.

Автентифікація пристрою. Кожен пристрій у мережі IoT повинен пройти процедуру автентифікації перед підключенням до шлюзу, щоб переконатися, що він не зламаний і йому можна довіряти.

Алгоритми шифрування. Щоб забезпечити високий рівень безпеки системи IoT, ви можете використовувати комбінацію симетричних і асиметричних алгоритмів шифрування. Наприклад, асиметричний алгоритм RSA і симетричний алгоритм Blowfish можуть бути добре реалізовані в апаратному забезпеченні IoT завдяки низькому енергоспоживанню.

Шифрування точка-точка. Дуже важливо шифрувати дані з моменту їх захоплення пристроєм Інтернету речей, доки ці дані не досягнуть точки дешифрування, наприклад шлюзу Інтернету речей або хмарного середовища.

Засоби безпеки для програмного забезпечення IoT

Безпека маршрутизації. Шифрування та хешування таблиць маршрутизації з даними, що зберігаються в маршрутизаторі, а також підтримка багатошляхової маршрутизації даних допомагають покращити безпеку даних IoT і підвищити відмовостійкість мережі IoT.

Захист даних користувача. Щоб уникнути несанкціонованого доступу до системи та забезпечити конфіденційність даних користувача, ви повинні встановити механізми автентифікації та перевірки особи у своєму програмному забезпеченні IoT.

Списки контролю доступу (ACL). Ще одним корисним рішенням для захисту додатків IoT є створення списків керування доступом (ACL), які містять політики та вказівки щодо дозволів для того, хто може отримати доступ до мережі IoT. Списки керування доступом можуть надавати доступ до системи або блокувати її для внутрішніх і зовнішніх користувачів системи.

Брандмауери. Додатковим заходом безпеки є встановлення брандмауерів. Це рішення використовується для блокування несанкціонованого доступу та спроб журналювання, якщо механізми автентифікації та ACL вийшли з ладу або зламані.

Програми захисту. Антивірусні та антишпигунські програми можуть бути додатковим заходом безпеки, який може врятувати систему IoT від потенційних зловмисних атак.

Наведений вище список елементів керування безпекою можна змінювати залежно від потреб кожного конкретного проекту IoT. Крім усіх засобів контролю безпеки, ви та ваша команда розробників програмного забезпечення повинні регулярно проводити сесії з оцінки ризиків і довірчого управління.

Аналітика та візуалізація даних IoT

Пропонувати послуги аналізу та візуалізації даних IoT необов'язково, але це може дати вам конкурентну перевагу. Однак такі послуги можуть вимагати більше зусиль від вашої команди розробників програмного забезпечення, оскільки звичайні інструменти бізнес-аналітики та аналітики можуть не підтримувати аналіз даних IoT через неструктурований характер даних IoT.

У цьому випадку ми віддаємо перевагу сервісу AWS IoT Analytics. Він автоматизує всі етапи аналізу даних IoT. Цей сервіс також підключається до Amazon QuickSight, який дозволяє візуалізувати дані за допомогою алгоритмів машинного навчання. Неструктуровані дані IoT надходять у необробленому форматі та можуть мати прогалини та помилкові показання, а AWS IoT Analytics очищає дані перед виконанням подальшого аналізу.

Ключові переваги служб аналізу даних IoT:

– Дані IoT не просто збираються, але й генерують уявлення та допомагають клієнтам краще побачити відчутну цінність їх мережі IoT.

– Збагачення аналізу даних IoT даними із зовнішніх джерел, наприклад прогнозів погоди, може допомогти клієнтам передбачити, як регулювати температуру вдома.

– Функціонал аналізу даних IoT також може дозволити клієнтам підвищити ефективність свого дому, бачачи закономірності в продуктивності домашніх пристроїв.

Останнім аспектом розробки програмного забезпечення для розумного дому є унікальний дизайн UI/UX, який може або привернути клієнтів, або відштовхнути їх.

UI/UX дизайн у розробці домашньої автоматизації

Користувачі очікують вдосконаленого дизайну інтерфейсу користувача/UX від рішень автоматизації розумного будинку. Завантажуючи нову цифрову банківську програму зі свого традиційного банку, користувачі можуть пробачити певні недоліки дизайну, якщо програма виконує важливі

банківські послуги. Але коли мова йде про програмне забезпечення для розумного дому, користувачі очікують чогось іншого та футуристичного. У той же час вони очікують, що додаток буде зручним і корисним.

Важливим аспектом, який відрізняє дизайн домашньої автоматизації IoT від проектування інших рішень, є прямий зв'язок між цифровим і фізичним світами. Розробникам потрібно знайти способи зробити так, щоб програмне забезпечення IoT полегшувало використання фізичних пристроїв IoT, а не викликало плутанину. Скевоморфний дизайн більше не є варіантом, але принаймні невелика схожість між цифровим і фізичним світом IoT повинна бути збережена.

Наприклад, у додатку, який ми розробили для нашого клієнта, дизайнери відобразили термостат так, як він виглядає у фізичному світі, щоб користувачі інтуїтивно зрозуміли, як збільшити або зменшити температуру в своєму домі. І, природно, вони вирішили позначити функцію управління світлом за допомогою лампочки.

Ми підготували кілька порад, які допоможуть вам створити належний дизайн інтерфейсу користувача/користувача користувача для програмного забезпечення IoT:

– Включіть чітку адаптацію користувача. Налаштування середовища IoT може бути проблемою для нетехнічних користувачів, тому ваше завдання – зробити їхню подорож максимально зрозумілою та простою. Ви можете включити захоплюючі аудіо- та візуальні елементи в дизайн вашої системи домашньої автоматизації, які направлятимуть користувачів через процес адаптації.

– Зробіть свій дизайн інклюзивним. Якщо припустити, що ви хочете надати свої продукти IoT якомога більшій кількості людей, вам потрібен інклюзивний дизайн. Наприклад, дуже важливо зробити ваше програмне забезпечення зрозумілим для людей із вадами зору або глухими.

– Додайте інтерактивні чат-боти для вирішення проблем. Щоб зменшити рівень розчарування користувачів, коли вони стикаються з проблемами своїх розумних домашніх пристроїв, ви можете створити інтерактивний чат-бот, який легко знайти. Приділяти особливу увагу розробці функцій усунення несправностей має вирішальне значення для програмного забезпечення IoT, оскільки простого екрана з інструкціями у звичайному тексті або просто кнопки для запиту підтримки може бути недостатньо.

– Вибирайте футуристичні кольори, форми та шрифти. Пристрої IoT асоціюються з технологіями майбутнього, тому було б чудово, щоб ваше програмне забезпечення виглядало футуристично, як ми зробили з додатком ConnectHome.

Поєднання технологій хмарних обчислень, належного контролю безпеки та привабливого дизайну UI/UX може допомогти вам розробити надійне програмне забезпечення IoT. Але обов'язково зосередьтеся також на апаратних компонентах, щоб ваші клієнти могли якомога плавніше поєднувати фізичний і цифровий світи. Обов'язково довірте свій IoT-проект надійному та досвідченому партнеру з розробки програмного забезпечення, який має продуктове мислення.

3.3 Розробка функціональної схеми

Modbus

У даному розділі розглянемо протокол Modbus. Протокол Modbus і мережа Modbus є найпоширенішими у світі. Незважаючи на свій вік (стандартом де-факто Modbus став ще в 1979 році), Modbus не тільки не застарів, але, навпаки, істотно зросла кількість нових розробок і обсяг організаційної підтримки цього протоколу.

Однією з переваг Modbus є відсутність необхідності в спеціальних інтерфейсних контролерах (Profibus і CAN вимагають для своєї реалізації замовлені мікросхеми), простота програмної реалізації й елегантність принципів функціонування. Все це знижує витрати на освоєння стандарту як системними інтеграторами, так і розроблювачами контролерного встаткування. Високий

ступінь відкритості протоколу забезпечується також повністю безкоштовними текстами стандартів, які можна скачати із сайту www.modbus.org.

На Україні Modbus по поширеності конкурує тільки з Profibus. Популярність протоколу в цей час пояснюється, насамперед, сумісністю з великою кількістю встаткування, що має протокол Modbus. Крім того, Modbus має високу вірогідність передачі даних, пов'язану із застосуванням надійного методу контролю помилок. Modbus дозволяє уніфікувати команди обміну завдяки стандартизації номерів (адрес) регістрів і функцій їх читання-запису. Основним недоліком Modbus є мережний обмін за типом "ведучий/ведений", що не дозволяє веденим пристроям передавати дані в міру їхньої появи й тому вимагає інтенсивного опитування ведених пристроїв ведучим. Різновидами Modbus є протоколи Modbus Plus – багатомастерний протокол з кільцевою передачею маркера й Modbus TCP, розрахований на використання в мережах Ethernet і інтернет. Протокол Modbus має два режими передачі: RTU (Remote Terminal Unit – «вилучений термінальний пристрій») і ASCII. Стандарт передбачає, що режим RTU у протоколі Modbus повинен бути присутнім обов'язково, а режим ASCII є опціонним. Користувач може вибирати кожний з них, але всі модулі, включені в мережу Modbus, повинні мати той самий режим передачі.

Ми розглянемо тільки протокол Modbus RTU, оскільки Modbus ASCII на Україні практично не використовується. Відзначимо, що Modbus ASCII не можна плутати із частково-фірмовим протоколом DCON, що використовується в модулях фірм Advantech і ICP DAS і не відповідає стандарту Modbus.

Стандарт Modbus передбачає застосування фізичного інтерфейсу RS-485, RS-422 або RS-232. Найпоширенішим для організації промислової мережі є 2-провідний інтерфейс RS-485. Для з'єднань точка-точка може бути використаний інтерфейс RS-232 або RS-422.

Таблиця 3.1 – Модель OSI для Modbus

Номер рівня	Назва рівня	Реалізація
7	Прикладний	MODBUS Application Protocol

6	Рівень подання	Немає
5	Сеансовий	Немає
4	Транспортний	Немає
3	Мережний	Немає
2	Канальний (передачі даних)	Протокол "ведучий/ведений" Режими RTU і ASCII
1	Фізичний	RS-485 або RS-232

У стандарті Modbus є обов'язкові вимоги, що рекомендуються й опціонні (необов'язкові). Існує три ступені відповідності стандарту: «повністю відповідає» – коли протокол відповідає всім обов'язковим і усім вимогам, які рекомендуються, «умовно відповідає» – коли протокол відповідає тільки обов'язковим вимогам і не відповідає що рекомендується, і «не відповідає».

Модель OSI протоколу Modbus містить три рівні: фізичний, канальний і прикладний.

Фізичний рівень

У нових розробках на основі Modbus стандарт рекомендує використовувати інтерфейс RS-485 із двопроводною лінією передачі, але допускається застосування чотирьохпроводною лінії й інтерфейсу RS-232.

Modbus-шина повинна складатися з одного магістрального кабелю, від якого можуть бути зроблені відводи. Магістральний кабель Modbus повинен містити 3 провідника в загальному екрані, два з яких являють собою кручену пару, а третій з'єднує загальні ("земляні") виводи всіх інтерфейсів RS-485 у мережі. Загальне проведення й екран повинні бути заземлені в одній точці, бажано біля провідного пристрою.

Пристрої можуть підключатися до кабелю трьома способами:

– безпосередньо до магістрального кабелю;

– через пасивний розгалужувач (трійник);

– через активний розгалужувач (утримуючий повторювач, що розв'язує, інтерфейсу).

У документації на пристрій і на трійник повинні бути зазначені найменування ланцюгів, що підключаються.

На кожному кінці магістрального кабелю повинні бути встановлені резистори для узгодження лінії передачі, як це потрібно для інтерфейсу RS-485. На відміну від фізичного інтерфейсу RS-485, у якому термінальні резистори на низьких швидкостях обміну можна не використовувати, стандарт на протокол Modbus формально вимагає застосування термінальних резисторів для всіх швидкостей обміну. Їхній номінал може бути рівним 150 Ом і потужність 0,5 Вт. Термінальні резистори, а також резистори, що усувають невизначеність стану лінії при високоомному стані передавачів, установлюються так само, як описано в розділі "Інтерфейси RS-485, RS-422 і RS-232". Стандарт вимагає, щоб у керівництвах з експлуатації пристроїв Modbus було сказано, чи є зазначені резистори усередині пристрою, або їх необхідно встановлювати при монтажі мережі. Якщо потрібні зовнішні резистори, то вони повинні мати номінал в інтервалі від 450 до 650 Ом і бути встановлені тільки в одному місці в межах кожного сегмента мережі (сегментами вважаються частини мережі між повторювачами інтерфейсу).

Modbus-пристрій обов'язково повинне підтримувати швидкості обміну 9600 біт/с і 19200 біт/с, з них 19200 біт/із установлюється "за замовчуванням". Допускаються також швидкості 1200, 2400, 4800,...,38400 біт/с, 65 кбіт/с, 115 кбіт/с,... .

Швидкість передачі повинна витримуватися в передавачі з погрішністю не гірше 1%, а приймач повинен приймати дані при відхиленні швидкості передачі до 2%.

Сегмент мережі, не утримуючий повторювачів інтерфейсу, повинен допускати підключення до 32 пристроїв, однак їхня кількість може бути збільшена, якщо це припустимо виходячи з навантажувальної здатності передавачів і вхідного опору приймачів, які повинні бути наведені в документації на інтерфейси. Вказівка цих параметрів у документації є обов'язковою вимогою стандарту.

Максимальна довжина магістрального кабелю при швидкості передачі 9600 біт/с і перетині жив більше 0,13 кв. мм (AWG26) становить 1 км. Відводи від магістрального кабелю не повинні бути більш довгою ніж 20 м. При використанні багатопортового пасивного розгалуджувача з N відводами довжина кожного відводу не повинна перевищувати значення 40 м/N.

Modbus не встановлює конкретних типів рознімань, але якщо використовуються рознімання RJ45, mini-DIN або D-Shell, вони повинні бути екранованими, а цоколівки повинні відповідати стандарту.

Для мінімізації помилок при монтажі рекомендується використовувати проведення наступних кольорів: жовтий – для позитивного виводу RS-485 (на якому встановлюється логічна "1", коли через інтерфейс виводиться логічна "1"); коричневий – для другого виводу інтерфейсу RS-485; сірий – для загального проведення.

Типовим перетином кабелю є AWG 24 (0,2 кв. мм, діаметр проведення 0,51 мм). При використанні кабелю категорії 5 його довжина не повинна перевищувати 600 м. Хвильовий опір кабелю бажано вибирати більше 100 Ом, особливо для швидкості обміну більше 19200 біт/с.

Канальний рівень

Протокол Modbus припускає, що тільки один провідний пристрій (контролер) і до 247 ведених (модулів уведення-виводу) можуть бути об'єднані в промислову мережу. Обмін даними завжди ініціюється ведучим. Ведені пристрої ніколи не починають передачу даних, поки не одержать запит від ведучого. Ведені пристрої також не можуть обмінюватися даними один з одним. Тому в будь-який момент часу в мережі Modbus може відбуватися тільки один акт обміну. Адреси з 1 по 247 є адресами Modbus пристроїв у мережі, а з 248 по 255 зарезервовані. Провідний пристрій не повинне мати адреси й у мережі не повинне бути двох пристроїв з однаковими адресами. Провідний пристрій може посилати запити всім пристроям одночасно ("широкомовний режим") або тільки одному. Для широкомовного режиму зарезервована адреса "0" (при використанні в команді цієї адреси вона приймається всіма пристроями мережі).

Опис кадру (фрейму) протоколу Modbus

У протоколі Modbus RTU повідомлення починає сприйматися як нове після паузи (тиші) на шині тривалістю не менш 3,5 символів (14 біт), тобто величина паузи в секундах залежить від швидкості передачі.

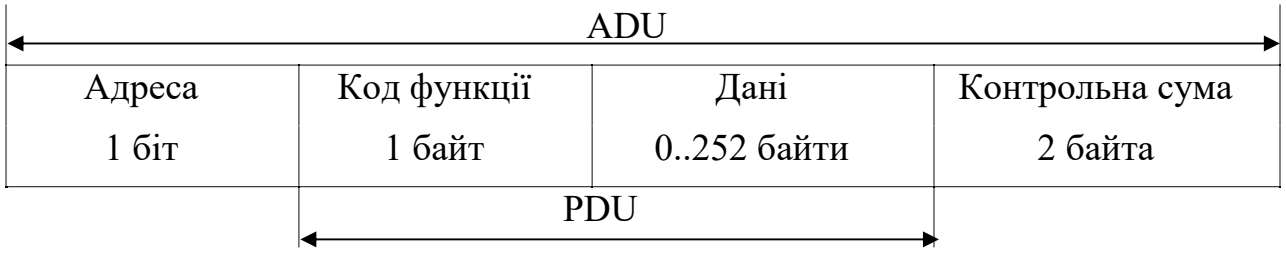

Рисунок 3.2 – Формат кадру протоколу Modbus RTU

На рисунку наведені наступні позначення:

PDU – "Protocol Data Unit" – "елемент даних протоколу";

ADU – "Application Data Unit" – "елемент дані додатки".

Формат кадру показаний на рис. 3.1. Поле адреси завжди містить тільки адреса веденого пристрою, навіть у відповідях на команду, послану ведучим. Завдяки цьому провідний пристрій знає, від якого модуля прийшла відповідь.

Поле «Код функції» говорить модулю про те, яке дія потрібно виконати.

Поле «Дані» може містити довільна кількість байт. У ньому може втримуватися інформація про параметри, використовуваних у запитах контролера або відповідях модуля. Поле «Контрольна сума» містить контрольну суму CRC довжиною 2 байти.

Структура даних у режимі RTU

У режимі RTU дані передаються молодшими розрядами вперед (рис. 3.2).

За замовчуванням в RTU режимі біт паритету встановлюють рівним 1, якщо кількість двійкових одиниць у байті непарне, і рівним 0, якщо воно парне. Такий паритет називають парним (even parity) і метод контролю називають контролем парності.

Стартовий біт	1 МЗР	2	3	4	5	6	7	8	Біт паритету	Стоп-біт

Рисунок 3.3 – Послідовність битов у режимі RTU

МЗР – молодший значущий розряд. При відсутності біта паритету на його місце записується другий стоп-біт

При парній кількості двійкових одиниць у байті біт паритету може бути дорівнює 1. У цьому випадку говорять, що паритет є непарним (odd parity).

Контроль парності може бути відсутнім взагалі. У цьому випадку замість біта паритету повинен використовуватися другий стоповий біт. Для забезпечення максимальної сумісності з іншими продуктами рекомендується використовувати можливість заміни біта паритету на другий стоповий біт. Ведені пристрої можуть сприймати кожний з варіантів: парний, непарний паритет або його відсутність.

Структура Modbus RTU повідомлення

Повідомлення Modbus RTU передаються у вигляді кадрів, для кожного з яких відомо початок і кінець. Ознакою початку кадру є пауза (тиша) тривалістю не менш 3,5 шістнадцяткових символів (14 біт). Кадр повинен передаватися безупинно. Якщо при передачі кадру виявляється пауза тривалістю більше 1,5 шістнадцяткових символа (6 біт), то вважається, що кадр містить помилку й повинен бути відхилений приймаючим модулем. Ці величини пауз повинні строго дотримуватися при швидкостях нижче 19200 біт/с, однак при більше високих швидкостях рекомендується використовувати фіксовані значення паузи, 1,75 мс і 750 мкс відповідно.

Контроль помилок

У режимі RTU є два рівні контролю помилок у повідомленні:

– контроль паритету для кожного байта (опційно);

– контроль кадру в цілому за допомогою CRC методу.

CRC метод використовується незалежно від перевірки паритету. Значення CRC установлюється в провідному пристрої перед передачею. При прийманні повідомлення обчислюється CRC для всього повідомлення й рівняється з його значенням, зазначеним у поле CRC кадру. Якщо обоє значення збігаються, уважається, що повідомлення не містить помилки.

Стартові, стопові біти й біт паритету в обчисленні CRC не беруть участь.

Прикладний рівень

Прикладний рівень Modbus RTU забезпечує комунікацію між пристроями типу "ведучий/ведений". Прикладний рівень є незалежним від фізичного й канального, зокрема, він може використовувати протоколи Ethernet TCP/IP (Modbus TCP/IP), Modbus Plus (багатомастерна мережа з передачею маркера), інтерфейси RS-232, RS-422, RS-485, оптоволоконі, радіоканали й інші фізичні середовища для передачі сигналів. Прикладний рівень Modbus заснований на запитах за допомогою кодів функцій. Код функції вказує веденому пристрою, яку операцію воно повинне виконати. При використанні протоколу прикладного рівня з різними протоколами транспортного й канального рівня зберігається незмінним основний блок Modbus-повідомлення, що включає код функції й дані (цей блок називається PDU – "Protocol Data Unit" – "елемент даних протоколу"). До блоку PDU можуть додаватися додаткові поля при використанні його в різних промислових мережах і тоді він називається "ADU" – "Application Data Unit" – "елемент дані додатки".

Коди функцій

Стандартом Modbus передбачені три категорії кодів функцій: установлені стандартом, що задаються користувачем і зарезервовані. Коди функцій є числами в діапазоні від 1 до 127. Коди в діапазоні від 65 до 72 і від 100 до 110 ставляться до задається користувачем, що, функціям, у діапазоні від 128 до 255 зарезервовані для пересилання кодів помилок у відповідному повідомленні. Код «0» не використовується. Коди помилок використовуються веденим пристроєм, щоб визначити, яку дію почати для їхньої обробки. Значення кодів і їхній зміст описані в стандарті на Modbus RTU .

Поле даних (рис. 3.1) у повідомленні, посланому від провідного пристрою веденому, містить додаткову інформацію, що ведене використовує, щоб виконати функцію, зазначену в поле «код функції». Поле даних може містити значення станів дискретних входів/виходів, адреси регістрів, з яких треба зчитувати (записувати) дані, кількість байт даних, посилання на змінні, кількість змінних, код підфункцій і т.п. Якщо ведений нормально виконав прийняту від ведучого функцію, то у відповіді поле «код функції» містить ту ж інформацію, що й у

запиті. У противному випадку ведений видає код помилки. У випадку помилки код функції у відповіді дорівнює коду функції в запиті, збільшеному на 128.

Зміст поля даних

У повідомленні провідного пристрою веденому поле даних містить додаткову інформацію, необхідну для виконання зазначеної функції. Наприклад, якщо код функції вказує, що необхідно вважати дані із групи регістрів пристрої уведення (код функції 03 hex), те поле даних містить адреса початкового регістра й кількість регістрів. Якщо провідний пристрій посилає команду запису даних у групу регістрів (код функції 10 hex), то поле даних повинне містити адреса початкового регістра, кількість регістрів, кількість байтів даних і дані для запису в регістр. Конкретний зміст поля даних установлюється стандартом для кожної функції окремо. У деяких повідомленнях поле даних може мати нульову довжину.

Список кодів Modbus

У табл. 3.2 наведений приклад кодів Modbus RTU для модуля дискретного уведення й виводу типу RealLab! NL-16DI (фірми НИЛ АП). Для читання логічних станів входів модуля через інтерфейс RS-485 необхідно послати команду у форматі, показаному на рис. 3.2 де в полях "Адреса", "Код" вказуються значення з відповідних граф табл. 3.2.

Таблиця 3.2 – Приклад кодів протоколу Modbus RTU для модуля RealLab! типу NL-16DI

Позначення регістра	HEX адреса регістра	Що читається або записується	Код функції читання регістра	Код функції запису в регістр	Примітка
1	2	3	4	5	6
00001	00h 00 h	Дискр. вихід 0	01	05	1 або 0
00002	00h 01 h	Дискр. вихід 1	01	05	1 або 0
10001	00h 00 h	Дискр. вхід 0	02	-	1 або 0
10002	00h 01 h	Дискр. вхід 1	02	-	1 або 0
10003	00h 02h	Дискр. вхід 2	02	-	1 або 0

Продовження таблиці 3.2

1	2	3	4	5	6
10004	00h 03h	Дискр. вхід 3	02	-	1 або 0
10005	00h 04h	Дискр. вхід 4	02	-	1 або 0
10006	00h 05h	Дискр. вхід 5	02	-	1 або 0
10007	00h 06h	Дискр. вхід 6	02	-	1 або 0
10008	00h 07h	Дискр. вхід 7	02	-	1 або 0
10009	00h 08h	Дискр. вхід 8	02	-	1 або 0
10010	00h 09h	Дискр. вхід 9	02	-	1 або 0
10011	00h 0Ah	Дискр. вхід 10	02	-	1 або 0
10012	00h 0Bh	Дискр. вхід 11	02	-	1 або 0
10013	00h 0Ch	Дискр. вхід 12	02	-	1 або 0
10014	00h 0Dh	Дискр. вхід 13	02	-	1 або 0
10015	00h 0Eh	Дискр. вхід 14	02	-	1 або 0
10016	00h 0Fh	Дискр. вхід 15	02	-	1 або 0
40201	00h C8 h	Ім'я модуля	03	10	
40213	00 h D4h	Версія програми	03	-	
40513	02h 00 h	Адреса модуля	03	06	0001 h-00 F7h (Припустимий діапазон значень)
40514	02h 01 h	Швидкість UART	03	06	0003 h-000 Ah (Припустимий діапазон значень)
40518	02h 05 h	Протокол	03	06	0000 h-h– ASCII, 0001h – RTU
40769	03h 00 h	Значення на виході після включення живлення модуля Power On Value0	03	06	0000 h-0003 h (Припустимий діапазон значень)

Групою пристроїв, які дають можливість організувати в системі домашньої автоматизації облік витрати ресурсів, є пристрої, які відносяться до класу серверів. Хоча, називатися у різних виробників вони можуть по-різному. З різними сенсорними панелями й екранами їх ріднить те обставина, що в якості операційної системи використовується Windows Server 2012. Виконавча частина програмного забезпечення зазвичай прив'язана до пристрою. До того ж, надійність таких пристроїв набагато вище, ніж у домашніх комп'ютерів. Вони відносяться до класу промислових комп'ютерів, зі всіма витікаючими вимогами до надійності.

Крім того, часто в них відсутні вентилятори охолодження, і використовуються тільки твердотільні накопичувачі, що робить їх роботу абсолютно безшумною.

Програмне забезпечення, що йде в комплекті, теж складається з двох частин. Сервісна частина може бути встановлена на будь-який комп'ютер під управлінням Windows. Розширений і зручний сервіс для роботи з додатками. Дуже часто на такому сервері підключення монітора не передбачено. Зате практично всі такі пристрої мають вбудовану мережну карту для підключення до мережі Ethernet. Деякі мають і вбудований Wi-Fi доступ.

Звернутися до такого серверу можна з будь-якого пристрою підтримуючого WEB-інтерфейс і має доступ до домашньої мережі. Домашня мережа повинна захищатися, та й сервера дозволяють організувати багаторівневу систему контролю доступу. Слід зазначити, що можна розробляти відразу кілька сторінок доступу. Дійсно, якщо ви зайшли на такий сервер з ноутбука, то отримуєте одну картинку і можливості, а з телефону з такою інформацією може виявитися працювати незручно, тому, там можна реалізувати мінімум функцій. Зрозуміло, якщо ваша домашня мережа має постійний доступ в інтернет, то вийти на сервер ви теж можете через інтернет, з будь-якої точки світу.

На рисунку 3.4 зображена функціональна схема системи розробленої для конкретного проекту, який відображений на структурній схемі. Нижче розглянемо її більш докладно.

Функціональні можливості системи розумний будинок

Управління системою

Тут реалізоване максимальне управлення системою: 4 DDP панелі+ 3 пульта управління до них. З кожної панелі можна керувати всіма споживачами в будинку + музикою, опаленням та кондиціонуванням. Крім того програмне забезпечення для двох комп'ютерних або портативних пристроїв. Наприклад одного ноутбука та одного планшета. Керувати розумним будинком з цих пристроїв можна з через Wi-Fi, а можна і з будь-якої точки планети, в якій є інтернет. Також двосторонній SMS шлюз може не тільки надсилати SMS а й отримувати до 24 видів SMS, кожне з яких виконувати до команд.

Освітлення

21 група освітлення. 2 групи кухня, 1 коридор, 1 ванні, 5 зал та 12 груп декоративної світлодіодної підсвітки, яку можна використовувати за вашим розглядом.(підсвітка картин, елементів інтер'єру, стелі і т.д.) На всі групи світла можна запрограмовувати сцени(наприклад «картинна галерія» – вимкнення всього світла крім підсвітки картин).

Аудіо-відео

Інфрачервоне управління телевізорами та плеєрами через багатофункціональні датчики. Крім того додано три модулі Z-audio, які можуть як працювати в парі, так і розділяти звук на різні зони. Можна прослуховувати музику, слухати оповіщення системи, а також віддалено керувати радіо з власного ФМ тюнера. Враховано вартість чотирьох колонок. Z-audio також має лінійний вхід, тому по команді може пускати в свої колонки звук з будь-якого інакшого джерела. Так само ви можете інфрачервоним зв'язком керувати відеоспостереженням, а на власному планшеті це переглядати. Можливе підключення системи мультирум, та керування нею з розумного дому, хоча доцільності в цьому в однокімнатній квартирі небагато.

Охорона, безпека

Захист від пожежі. В кожній кімнаті змонтованй датчик диму. При спрацюванні, відключається електроенергія, крім основних споживачів.

Вмикається голосове оповіщення «Увага дим». Власнику надсилається повідомлення «Спрацювання датчика диму». Види реакцій на будь-яку аварійну ситуацію логічний модуль може розрахувати, якщо в нього запрограмувати відповідні функції. Наприклад, якщо квартира не під охороною(власник дома)система не буде надсилати SMS. Захист від газу – Датчик газу на кухні. При спрацюванні перекривається клапан газу в квартирі, вмикається витяжна вентиляція. Вмикається голосове оповіщення «Увага витік газу». Власнику надсилається SMS «Спрацювання датчика газу».

Захист від протікання води. В ванній та в кухні на підлозі розміщені датчики води. При потрапляння на них води, реле вмикає електромагнітний клапан і він перекриває центральну подачу води в квартирі. Вмикається голосове оповіщення «Увага витік води» Власнику надсилається SMS «Спрацювання датчика води».

Охорона. Датчики руху в кожному приміщенні, геркони на дверях і вікнах, можливість постановки або зняття з сигналізації з будь-якої DDP панелі

Клімат

В даному випадку розумний будинок бере керування кліматом повністю під власний контроль. DDP панелі стежитимуть за температурою в приміщеннях та керуватимуть теплою підлогою, або електромагнітними клапанами на радіаторах, кліматичний модуль повністю візьме на себе управління кондиціонером. Логічний модуль буде стежити за їх правильною взаємодією, та взаємодією з розумним будинком загалом

Інші прилади

Вентилятор в ванній, на кухні. Ввімкнення вимкнення вентиляторів можна налаштувати на ваш смак: по датчиках руху, закриванні – відкриванні дверей, ввімкненні – вимкненні світла, по часу доби, дню тижня, задати бажаний час роботи з подальшим відімкненням.

Розетки. Керуючи розетками ви можете керувати будь-яким приладом, яким й в неї ввімкнений.

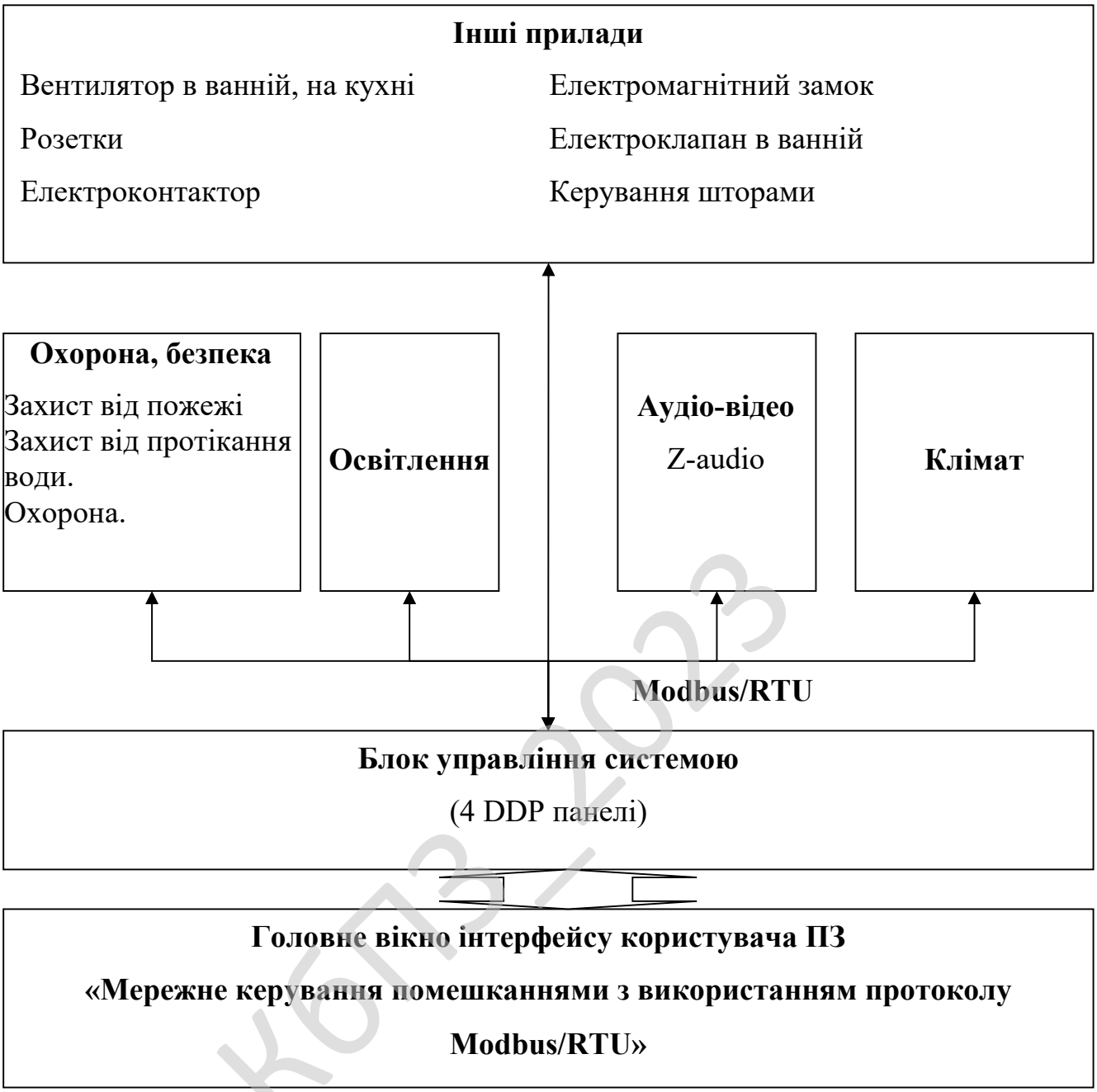

Рисунок 3.4 – Функціональна схема системи

Якщо ви ввімкнете в неї опалювальний прилад і відповідно запрограмуєте DDP панель, то вона буде підтримувати задану вами температуру в приміщенні.

Електроконтактор в щиті. Керуючи ним, ви керуєте всіма споживачами, які до нього під'єднані. Найчастіше до нього під'єднують всі розетки. Можливості управління. Наприклад він може відключити всі розетки при спрацювання датчика диму, щоб унеможливити потраплянню під дію електричного струму.

Електромагнітний замок – виконує функції безпеки.

Електроклапан в ванній – по вашій команді, або по таймеру(наприклад за 10 хв. до ввімкнення будильника) набере воду в ванній. Головне виробити звичку закривати стічний отвір. Вимикатись може по датчику води, який ви розмістите у ванній або просто по таймеру.

Керування шторами – Реле Smart Bus дозволяє керування електродвигунами. Для цього передбачене блокування парних контактів (при ввімкненні одного реле не можна ввімкнути інше), до яких ми і під'єднуємо двигуни придбаних вами карнізів. Керувати ними можна з будь яких приладів управління, а логічний модуль може навіть пов'язати це із сходом та заходом сонця.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання дипломного проектування, наведена на рисунку 3.5.

Після початку роботи системи проводиться виведення головного вікна ПЗ далі можна переглянути довідкову систему та авторське право ПЗ чи через підключення бібліотеки Modbus/RTU проводити моніторинг чи керування.

Моніторинг даних проводиться з отриманням даних клімат контролю, отримання даних освітлення, стану системи та датчиків та отримання даних охоронної системи.

Керування проводиться через блок управління системою та може проводити керування освітленням, керування системою безпеки, керування параметрами клімат контролю, керування параметрами аудіо-відео, керування додатковими приладами.

Рисунок 3.5 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення головного вікна розробленого ПЗ. Після цього проходе виконання основного коду який складається з наступних блоків:

– Ініціалізація.

– Налаштування.

– Опит систем керування.

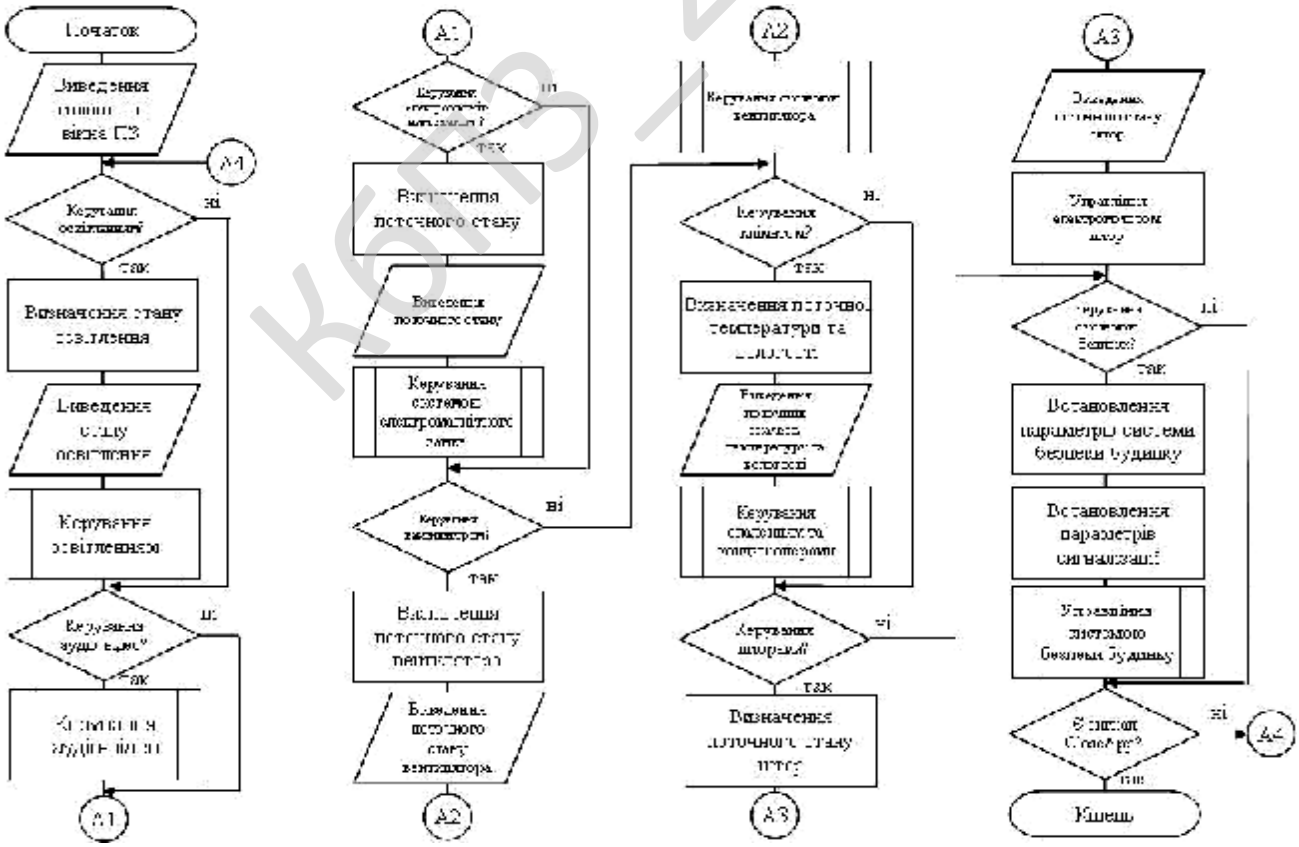

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено блок-схема підпрограми моніторингу – керування пристроєм. Як можна побачити з блок-схеми вона розділяється на блоки – підключення, обчислення та обробки команд.

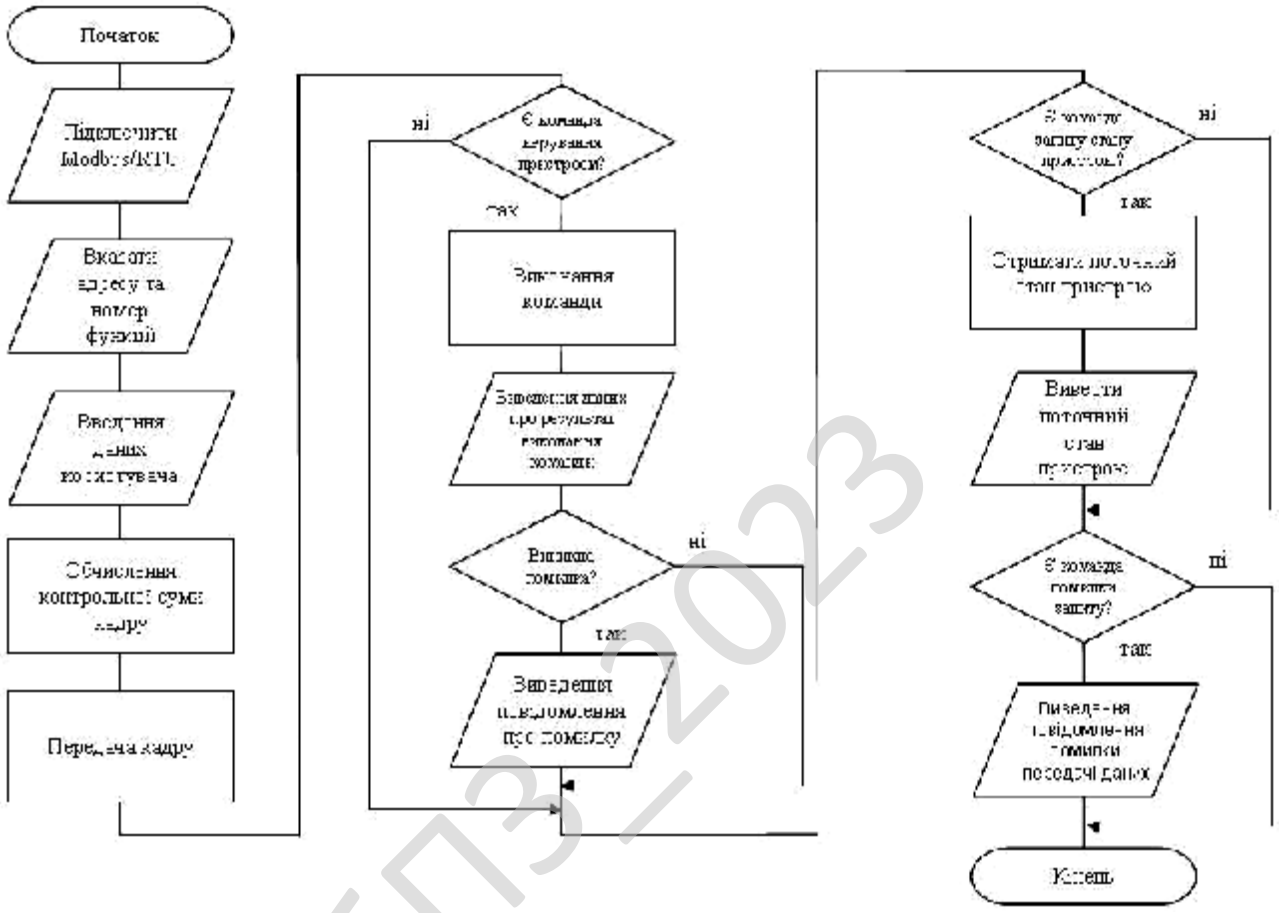

Рисунок 4.2 – Блок-схема підпрограми моніторингу

Реалізація протоколу Modbus

Що ж таке цей Modbus – це протокол, заснований на взаємодії клієнта і сервера шляхом запитів і відповідей. Біля витоків розробки Modbus в далекому 1979 році варто фірма Modicon, яка застосовувала даний протокол у своїх контролерах з програмованою логікою.

Зараз Modbus став вже де-факто стандартом, був перекуплений Schneider Electric і став широко застосовуватися в промисловості для організації зв'язку між найрізноманітнішими приладами та програмами.

Протокол може використовувати для передачі даних як послідовні лінії зв'язку RS-485, RS-422, RS-232, так і мережі TCP/IP, що значить, і Інтернет.

Багато виробників електронного устаткування підтримали стандарт, і в результаті на ринку з'явилися сотні використовують його виробів. У теперішній же час розвитком Modbus займається також некомерційна організація Modbus-IDA.

Розрізняють такі різновиди Modbus-протоколу:

– Modbus RTU (для передачі чисел).

– Modbus ASCII (для передачі символів).

– Modbus TCP (Modbus рівень Ethernet, в мережі Інтернет).

– Modbus Plus (мультімастерний пропріетарний протокол).

Як правило, в MODBUS мережі є лише один головний клієнт, званий master, і кілька підлеглих серверів, які називаються slave.

MASTER пристрій посилає запити, а підлеглі виконують їх, передають дані і відповідають про свою готовність. MASTER може звертатися як індивідуально до підлеглого і чекати відповіді, так і передавати широкомовний запит для всіх пристроїв в мережі.

Відповідь також починається з адреси відповідає веденого пристрою, який може змінюватися від 1 до 247.

Адреса "0" використовується для широкомовної передачі. У разі отримання широкомовного запиту відповідь повідомлення підлеглими не формується. Сам же елементарний пакет запиту в протоколі включає в себе код функції, поле даних і блок виявлення помилок і обмежений довжиною в 253 байта.

Код функції кодується однобайтового полем і може приймати значення в діапазоні 1...127.

Діапазон значень 128...255 зарезервований для кодів помилок. Поле даних може бути змінної довжини.

Стандартні функції протоколу Modbus

Специфікація протоколу безпосередньо визначає наступні категорії кодів функцій:

– Користувальницькі (діапазон кодів функцій від 65-72 і від 100-110);

– Зарезервовані (9, 10, 13, 14, 41, 42, 90, 91, 125, 126 і 127).

Для читання декількох значень використовуються функції з кодами з 1 по 4:

– 0x01 читання значень з декількох регістрів прапорів;

– 0x02 читання значень з скількох дискретних регістрів;

– 0x03 читання значень з декількох регістрів зберігання;

– 0x04 читання значень з декількох регістрів введення.

Для запису одного значення:

– 0x05 – запис значення одного прапора;

– 0x06 – запис значення в один регістр зберігання.

Команда складається з адреси елемента (2 байта) і встановлюваного значення(2 байти). Для регістру зберігання значення є просто 2-х байтним словом. Для прапорів значення 0xFF00 означає включений стан, 0x0000 – вимкнене, інші значення неприпустимі.

Якщо команда виконана успішно, SLAVE пристрій повертає копію запиту. Розглянемо приклад команди ведучого пристрою і відповіді SLAVE для протоколу Modbus. Контроль помилок в протоколі Modbus RTU.

При будь-якому обміні даними можуть виникнути помилки двох типів:

– Помилки, пов'язані з спотвореннями при передачі даних по лінії зв'язку;

– Логічні помилки.

Помилки першого типу легко виявляються за допомогою контролю парності і циклічної контрольної суми (CRC-16 з числом полиномом 0xA001).

Моніторинг

Як ми вже знаємо, для проведення діагностики в протоколі MODBUS існує спеціально виділена функція з кодом 0x08.

Функція "08" забезпечує так звану серію тестів для перевірки системи зв'язку між головним (master) і підлеглим (slave) або перевірку на різні внутрішні помилки в підлеглому.

Одночасне мовлення не підтримується, і це логічно зрозуміло, адже одночасний відповідь безлічі пристроїв у мережі неприпустимий.

Для передачі даних в сторонні додатки доцільно введення до складу монітора – OPC (OLE for Process Control) сервера, що дозволить надалі надавати опис даних і функціональних можливостей контрольованого приладу в стандартизованому вигляді і здійснювати обмін з ним за запитами будь-якої автоматизованої системи, наприклад SCADA.

OPC дуже схожий з протоколом MODBUS, оскільки заснований теж на клієнт-серверній архітектурі.

Розглянемо приклади розробленого коду взаємодії.

Проект розробленої програми (Файл Project_IntellectHome_ Modbus_RTU.cpp):

```
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------
USEFORM("main.cpp", Form_main);
// Головне вікно
USEFORM("about.cpp", Form_about);
// Вікно даних про розробника
USEFORM("light.cpp", Form_light);
// Вікно управління світлом
USEFORM("water.cpp", Form_water);
// Вікно управління водою
USEFORM("gas.cpp", Form_gas);
// Вікно управління газом
USEFORM("security.cpp", Form_security);
// Вікно управління захистними системами
USEFORM("climate.cpp", Form_climate);
// Вікно управління кліматом
USEFORM("phone.cpp", Form_phone);
// Вікно управління телефонією
USEFORM("tv.cpp", Form_tv);
// Вікно управління телевізорами
```

```
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                 Application->Initialize();
// Створення головної форми
                 Application->CreateForm(__classid(TForm_gas), &Form_gas);
// Створення форми управління захистними системами
Application->CreateForm(__classid(TForm_security), &Form_security);
// Створення форми управління кліматом
        Application->CreateForm(__classid(TForm_climate), &Form_climate);
// Створення форми управління телефонією
               Application->CreateForm(__classid(TForm_phone),
               &Form_phone);
// Створення форми управління телевізорами
                Application->CreateForm(__classid(TForm_main),
               &Form_main);
// Створення форми даних про розробника
                Application->CreateForm(__classid(TForm_about),
               &Form_about);
// Створення форми управління світлом
                Application->CreateForm(__classid(TForm_light),
               &Form_light);
// Створення форми управління водою
                Application->CreateForm(__classid(TForm_water),
               &Form_water);
// Створення форми управління газом
                Application->CreateForm(__classid(TForm_tv), &Form_tv);
// Створення форми запуску проекту
                Application->Run();
        }
        catch (Exception &exception)
        {
                 Application->ShowException(&exception);
        }
        catch (...)
        {
                 try
                 {
                         throw Exception("");
                 }
```

```
                        catch (Exception &exception)
                        {
                                Application->ShowException(&exception);
                        }
                }
                return 0;
        }
```

Розшифрування пакету MODBUS та перевірка контрольної суми:

```
        void __fastcall TCPU_Modbus_RTU_Socket::MyOnRead(System::TObject* Sender,
TCustomWinSocket* Socket)
        {

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

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

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

        NewLoop:

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

        if (( size-Offset)<2)
```

```
    goto end_label;
else
    {
    memcpy(&CommandStrSize,local_pointer,2);
// Довідалися довжину текстової команди
    Offset=Offset+2;
    local_pointer=&buff[Offset];
    }
if (( size-Offset)<CommandStrSize) goto end_label;
// Якщо шматок до кінця залишився менше,
// ніж довжина команди, виходимо з функції
switch ( local_pointer[0] )
{
case  'a':
            if (CommandStrSize==2)
                {
                FAuth=local_pointer[1];
                if (FOnChangeAuthStatus)
                    FOnChangeAuthStatus(this,FAuth);
                };
            break;
case  'c': break;

case  'd':
            if (CommandStrSize==2)
                {
                FAuth=2;
                if (FOnChangeAuthStatus) FOnChangeAuthStatus(this,FAuth);
                };
            break;
case  'p':
if (FOnReadOtherData)
                FOnReadOtherData(this,&local_pointer[0],CommandStrSize);

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

```
                    if (CommandStrSize==3)
                        if (FOnReadOcelotStatusData)
                            FOnReadOcelotStatusData(this,false);
                    break;

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

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

        end_label:
        free(buff);
        }
        //------- -------------------
```

Категорії кодів функцій

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

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

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

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

Модель даних

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

Таблиця 4.1 – Таблиця даних

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

Доступ до елементів в кожній таблиці здійснюється за допомогою 16-бітного адреси, першій клітинці відповідає адреса 0. Таким чином, кожна таблиця може містити до 65536 елементів.

Специфікація не визначає, що фізично повинні представляти собою елементи таблиць і по яким внутрішнім адресами пристрою вони повинні бути доступні.

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

Рисунок 4.3 – Стандартні функції протоколу Modbus

Наприклад, припустимо організувати перекриваються таблиці, У цьому випадку команди працюють з дискретними даними і з 16-бітними регістрами будуть фактично звертатися до одних і тих же даних.

де:

– A_1 і A_0 – адреса елемента;

– Q_1 і Q_0 – кількість елементів;

– N – кількість байт даних;

– D – дані.

Слід зазначити, що зі способом адресації даних пов'язана певна плутанина. Modbus був спочатку розроблений для контролерів Modicon. У цих контролерах для кожної з таблиць використовувалася спеціальна нумерація. Наприклад, першому регістру введення відповідав номер комірки 30001, а першому регістру зберігання – 40001.

Таким чином, регістру зберігання з адресою 107 в команді Modbus відповідав регістр № 40108 контролера. Хоча така відповідність адрес більше не є частиною стандарту, деякі програмні пакети можуть автоматично «коригувати» вводяться користувачем адреси, наприклад, віднімаючи 40001 з адреси регістра зберігання.

Читання даних

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

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

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

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

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

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

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

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

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

Рисунок 4.4 – Значення регістрів зберігання і регістрів введення

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

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

Рисунок 4.5 – Значення прапорів і дискретних входів

Запис одного значення:

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

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

Команда складається з адреси елемента (2 байти) і встановлюваного значення (2 байти). Для регістру зберігання значення є просто 16-бітним словом. Для прапорів значення 0xFF00 означає включений стан, 0x0000 – вимкнений, інші значення неприпустимі. Якщо команда виконана успішно, ведене пристрій повертає копію запиту. Запис декількох значень:

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

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

Команда складається з адреси елемента, кількості змінюваних елементів, кількості переданих байт встановлюваних значень і самих встановлюваних

значень. Дані упаковуються так само, як в командах читання даних. Відповідь складається з початкової адреси і кількості змінених елементів. Нижче наведено приклад команди ведучого пристрою і відповіді веденого (для Modbus RTU).

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

Рисунок 4.6 – Приклад команди ведучого пристрою

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

Рисунок 4.7 – Приклад відповіді веденого пристрою

Контроль помилок у протоколі Modbus RTU

Під час обміну даними можуть виникати помилки двох типів:

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

– Логічні помилки.

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

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму SHA-3 (Keccak) – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Keccak став переможцем конкурсу криптографічних алгоритмів, проведеним Національним інститутом стандартів і технологій США. 5 серпня 2015 року алгоритм затверджено та опубліковано в якості стандарту FIPS 202[1]. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Проте в апаратних реалізаціях Keccak виявився набагато швидшим, ніж всі інші фіналісти.

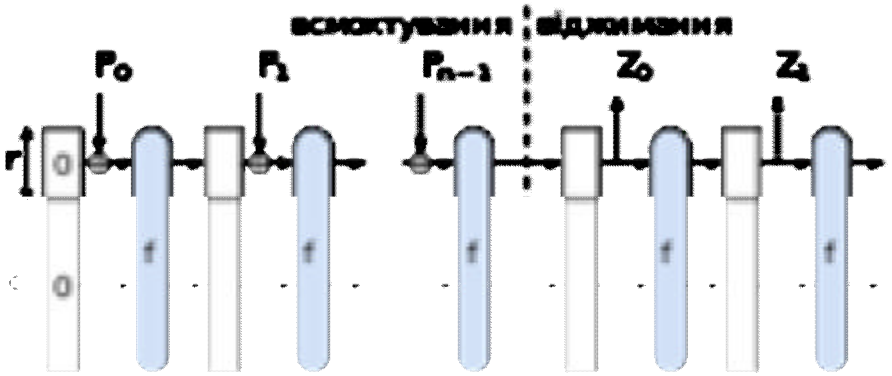

Рисунок 4.8 – Конструкція функції губки, використана в геш-функції

Конструкція функції губки, використана в геш-функції. P_i – вхідні блоки, Z_j – вихід алгоритму. Невикористаний для виведення набір бітів c («capacity») повинен мати значний розмір для досягнення стійкості до атак.

Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak раніше).

Геш-функції сімейства SHA-3 побудовані на основі конструкції криптографічної губки, в якій дані спочатку «вбираються» в губку, при якому початкове повідомлення M піддається багатораундовим перестановкам f, потім результат Z «віджимається» з губки. На етапі «вбирання» блоки повідомлення додаються за модулем 2 з підмножиною стану, який потім перетвориться з допомогою функції перестановки f. На етапі «віджимання» вихідні блоки зчитуються з одного і того ж підмножинного стану, зміненого функцією перестановок f. Розмір частини стану, який записується і зчитується, називається «швидкістю» (англ. rate) і позначається r, а розмір частки, яка незаймана введенням / виведенням, називається «ємністю» (англ. capacity) і позначається c.

Алгоритм отримання значення хеш-функції можна розділити на кілька етапів:

– Вихідне повідомлення M додається до рядка P довжини, кратній r, за допомогою функції доповнення (pad-функції).

– Рядок P ділиться на n блоків довжини r: P_0, P_1,..., P_{n-1}

– «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжини b біт і підсумовується по модулю 2 з рядком стану S, де S – рядок довжини b біт (b = r + c). Перед використанням цієї функції всі елементи S дорівнюють нулю. Для кожного наступного блоку стан – рядок, отриманий застосуванням функції перестановок f до результату попереднього кроку.

– «Віджимання»: поки довжина Z менша d (d – кількість біт в результаті геш-функції), до Z додається r перших біт стану S, після кожного додавання до S, застосовується функція перестановок f. Потім S обрізається до довжини d біт.

– Рядок Z довжини d біт повертається в якості результату.

Завдяки тому, що стан містить c додаткових біт, алгоритм стійкий до атаки подовженням повідомлення, до якої прийняті алгоритми SHA-1 і SHA-2.

У SHA-3 стан S – це масив 5×5 слів довжиною w = 64 біта, всього $5 \times 5 \times 64 = 1600$ біт. Також в Keccak можуть використовуватися довжини w, рівні меншим ступенями 2 (від w = 1 до w = 32).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Основне меню ПЗ.

– Моніторинг стану об’єктів керування.

– Функціонального блоку керування.

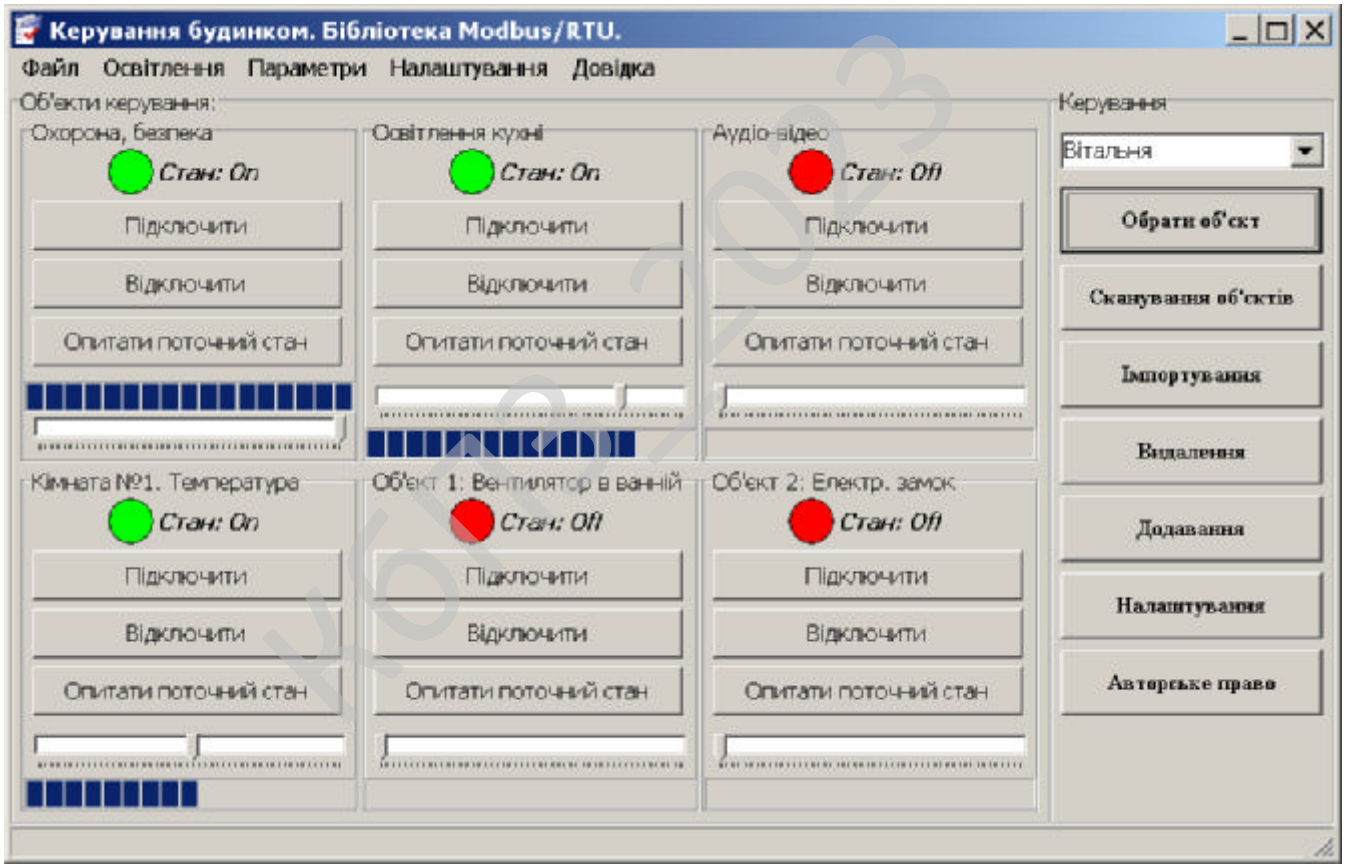

Рисунок 5.1 – Головне вікно програми

У свою чергу меню складається з розділів:

– Файл.

– Освітлення.

– Параметри.

– Налаштування.

– Довідка.

Об’єкти керування складаються:

– Охорона, безпека.

– Освітлення кухні.

– Аудіо-відео.

– Кімната №1. Температура.

– Об'єкт 1: Вентилятор в ванній.

– Об'єкт 2: Електричний замок.

Блок керування складається: Назви обраної групи; Обрання об'єкту; Сканування об'єктів; Імпортування; Видалення; Додавання; Налаштування; Авторське право.

На рисунку 5.2 зображено форму авторського права.

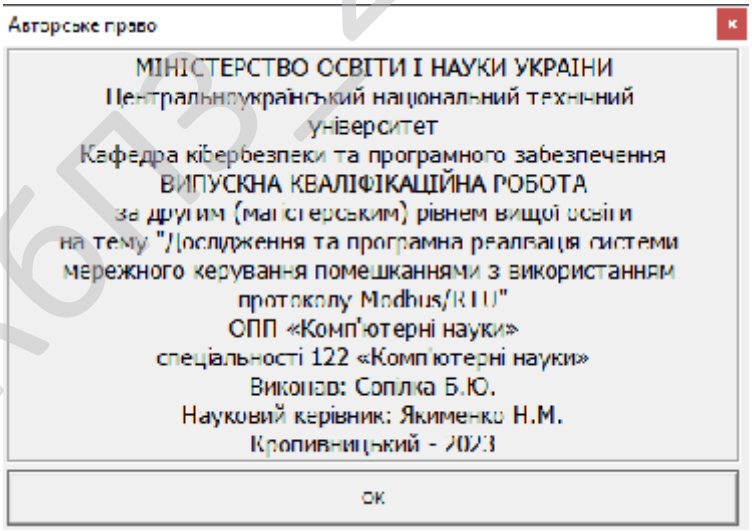

Рисунок 5.2 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Метою розробки є дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Об'єктом дослідження є процес мережного керування помешканнями з використанням протоколу Modbus/RTU.

Предметом дослідження є методи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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 мережного керування помешканнями з використанням протоколу Modbus/RTU.

– Розроблено вітчизняний продукт мережного керування помешканнями з використанням протоколу Modbus/RTU, 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 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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	22
3. Запланований термін розробки, днів	Fpq	60 (3 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	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
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)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
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)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	22000
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:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

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

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

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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:

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

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

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 70 = 117 \text{ люд/день.}$$

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	9	Д7
Робочий проект	117	Ф 7.1-7.4
Впровадження	13	Д13
Всього	158	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{158 \cdot 1}{60 - 5} = 2{,}8 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	280	700	11,67
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	53,33

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф^{с}_{др} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф^{с}_{др} = \frac{53 \cdot 3}{1,2} = 132,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 132,5/(60·8) = 0,27 ставки.

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

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

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

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

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	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	18132	54396
Продакт-менеджер	0,25	14000	10500
Інженер-програміст	2,8	18000	151200
Інженер-електронщик	0,27	14000	11340
Інженер-системотехнік	0,25	14000	10500
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 6,32	-	$Ф_{роб}$ = 301086

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.

$$З_{сд} = \frac{301086}{6{,}32 \cdot 60} = 794 \text{ грн.}$$

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

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

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

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

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

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

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

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

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу Supercomp за 25.10.23 – джерело https://supercomp.kiev.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11186
Системний блок		6490
Процесор	AMD Ryzen 3 2200G PRO, AM4, 4 ядра, потоки, 3.5 GHz, 3.7 GHz, TDP – 65 Вт 14nm, L1: 384KB, L2: 2MB, L3: 4MB Radeon Vega 8, Zen, Tray	-
Системна плата	Biostar A520MH 3.0, AM4, DDR4, 64 ГБ 4400 MHz, LAN – 1 Гбіт/с, D-Sub (VGA) HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s Micro-ATX	-
Відеокарта	вбудована	-
Жорсткий диск	SSD M.2 2280 240GB Apacer, 3D TLC, PC Express 3.0 x4, Читання – 1800 Mb/s, Запис 1100 MB/s, 80 x 22 x 2.25 мм	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patric (PSD48G240081) DDR4, 8 ГБ, В наборі – 1	-
Система охолодження	Vinga CL3011 Для процесорів – INTEL AMD, сокет – 1151, AM4, 1150, 1155, AM2 AM3, AM3+, 775, 1156	-
Корпус	Vinga CS104B-500W, Miditower, ATX Micro – ATX, Mini – ITX, PSU – 500 Вт Слотів – 7, – Back 1x12 см, Front 1x12 см GPU – до 270 мм, PSU – 200 мм, 180 x 410 410 мм, чорний	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3200
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1496

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни. Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	11186	8948,8	98436,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	8948,8	98436,8
Копіюв. апарат	1	5965	540	5940
Всього	–	–	–	214803,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	214804	-	-
Всього по групі	214804	50	107402
Група 5, 6			
4. Вимірювальні пристрої	5190	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	28000	25	-
Всього по групі	33190	-	8297,5
Нематеріальні активи			
7. Нематеріальні активи	22000	10	2200
Разом	$К_р$ = 1677994		$А_р$ = 188299,5

Примітка: вартість автомобіля приймаємо рівною нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 794 \cdot 158/22 = 5700 \text{ грн.}$$

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

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

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

$$З_д = 5700 \cdot 10 \cdot 0{,}01 = 570 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(5700+570) = 1379 \text{ грн.}$$

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, %.

$$Г_{осп} = 5700 \cdot 15 \cdot 0{,}01 = 855 \text{ грн.}$$

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

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

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,75 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0,75 = 150 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 27,6 грн./шт., DVD-R box – 32,15 грн./шт.

$$З_{М2} = 10 \cdot 27,6 = 276 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (150 + 276 + 1702)/22 = 97 \text{ грн.}$$

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, %.

$$О_п = 5700 \cdot 15 \cdot 0,01 = 855 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 22 прим.):

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

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

$$A_м = 188300 \cdot 3/(22 \cdot 12) = 2140 \text{ грн.}$$

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

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

$$C_п = 5700+570+1379+855+97+855+2140 = 11596 \text{ грн.}$$

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%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 11596 = 5798 \text{ грн.}$$

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	$З_о$	5700
2. Додаткова зарплата виконавців	$З_д$	570
3. Відрахування на соціальні потреби	$С_{оц}$	1379
4. Загальногосподарські витрати	$Г_{осп}$	855
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	855

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	2140
8. Повна собівартість програмного забезпечення	$С_п$	11596
9. Плановий прибуток	$П_р$	5798
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	17394
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3478,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	20872,8

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	20873
Всього капітальних витрат	–	20873

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	$З_р$	60390	42273
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	5218
Всього витрат за рік	$І$	60390	47491

Витрати на обслуговування:

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

де: $Т_р$ – кількість годин, що витрачається на обслуговування на рік, *год.* (приймаємо 500 год.);

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

$$З_{р\ баз} = 500 \cdot 90 \cdot 1{,}1 \cdot 1{,}22 = 60390\ грн,$$

до:

$$З_{р\ нов} = 350 \cdot 90 \cdot 1{,}1 \cdot 1{,}22 = 42273\ грн.$$

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 ($Ц_{ел}$):

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

$$З_{ел\ баз} = З_{ел\ нов}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	20873	–	5218,25
Всього відрахувань	-	–	20873	–	5218,25

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:

$$E_в = (Ц_п - С_п) \cdot N_в - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

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

$$Ев = (17394 - 11596) \cdot 22 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 214804 + +0{,}25 \cdot 33190 + 0{,}1 \cdot 22000) \frac{\cdot 3}{12} = 80481 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - С_п) \cdot N_в}, \quad (7.26)$$

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

$$T_в = \frac{1677994}{(17394-11596) \cdot 21 \cdot 12/3} = 3{,}2 \text{ року.}$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	22
2. Повна собівартість розробленої програми	Грн.	11596
3. Ціна розробленої програми	Грн.	17394
4. Плановий прибуток від реалізації одної програми	Грн.	5798
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1677994
7. Загальний прибуток від реалізації програмної продукції	Грн.	127556
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	80481
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	3,2
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	20873
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7681
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	1,62

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (60390\text{-}47491)\text{-}0{,}25 \cdot 20873 = 7681 \text{ грн.}$$

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

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

$$T_{сп} = \frac{20873}{60390 - 47491} = 1{,}62 \; року.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

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

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

Ці шкідливі фактори можуть привести до професійних захворювань.

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

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Це важливе питання, адже робота з комп'ютером може впливати на здров'я людини. За ДСанПіН 3.3.2.007-98 шкідливі і небезпечні фактори при роботі з комп'ютером можуть бути такі:

– Електромагнітне випромінювання – це випромінювання, яке створюється комп'ютером і його периферійними пристроями, такими як монітор, пртер, сканер тощо. Це випромінювання може викликати головний біль, запаморчення, розлади сну, зниження імунітету та інші негативні ефекти.

– Електростатичне поле – це поле, яке утворюється внаслідок накопичення електричних зарядів на поверхні комп'ютера і його частин. Це поле може спричиняти сухість шкіри, свербіж, подразнення очей, алергічні реакції та інші проблеми.

– Нервово-емоційне напруження – це напруження, яке виникає внаслідок тривалої концентрації уваги, високої вимогливості до результату роботи, нестабільності програмного забезпечення, конфліктних ситуацій тощо. Це напруження може призводити до стресу, депресії, роздратування, погіршення пам'яті та інших порушень.

– Навантаження на органи зору – це навантаження, яке виникає внаслідок тривалого перебування перед монітором, низької якості зображення, недостнго

освітлення приміщення, поганої ергономіки робочого місця тощо. Це навантаження може спричиняти зниження гостроти зору, появу блимавок і кругів перед очима, синдром сухого ока та інші захворювання.

– Дрібні стереостатичні рухи кінцівок – це рухи, які пов'язані з керуванням клавіатурою і мишею. Ці рухи можуть призводити до перевантаження і запалення суглобів і сухожиль рук і пальців, а також до тендовагінітів і синдрому зап'ясткового каналу.

Таким чином, робота з комп'ютером не така безпечна, як може здатися н перший погляд. Тому дуже важливо дотримуватися правил охорони праці і гігієн при роботі з комп'ютером.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	6
Довжина	6
Висота	3,2

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

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значенн
Площа, S	$м^2$	не менше 6.0	6
Обсяг, V	$м^3$	не менше 20.0	19,2

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

У зазначеному приміщенні працює 6 людей. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], а об'єм – НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

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

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

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	21,8-23	40-60	0,1
Тепла	23-25	50-70	0,1	23-25	52-70	0,11

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

В літню пору застосовується кондиціонер.

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

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

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

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

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних 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).

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 6 м.; довжина –6 м.; висота – 3,2 м.

Розрахунок штучного освітлення проведемо за методом коефіцієнта використання світлового потоку.

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

$$F=ESKZ/n,$$

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

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

S – площа освітлюваного приміщення (у нашому випадку *S*=6 х 6 = 36 $м^2$);

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

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

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

Обчислимо індекс приміщення за формулою:

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

де: *S* – площа приміщення, *S = 36* м^2;

h – розрахункова висота підвісу, *h* = 3,2 *м;*

A – ширина приміщення, А = 6 м;

B – довжина приміщення, В = *6* м.

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

Знаючи індекс приміщення, за знаходимо *n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу).* Підставимо всі значення у формулу, визначемо світловий потік: *F=38739 Лм.*

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

Число світильників визначається по формулі:

$$N=F/F_л$$

де: *F* – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 38739/ 8000=4,8 \text{ шт.}$$

Приймаємо необхідну кількість світильників 5 шт.

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

Список використаних джерел інформації

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

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

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

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

5. 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

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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 мережного керування помешканнями з використанням протоколу Modbus/RTU.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мережного керування помешканнями з використанням протоколу Modbus/RTU.

– Досліджена система мережного керування помешканнями з використанням протоколу Modbus/RTU.

– На основі отриманих результатів досліджень створена програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання мережного керування помешканнями з використанням протоколу Modbus/RTU.

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 SHA-3.

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 7681 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,62 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сопілка Б.Ю. Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

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

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

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

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

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

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

9. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools».

Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в

інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU.

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 мережного керування помешканнями з використанням протоколу Modbus/RTU;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 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 – 129 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

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

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Якименко Н.М.

Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 47

Літера: РП

Кропивницький – 2023 року

Основна програма

Файл Project_IntellectHome_Modbus_RTU.cpp основної програми

```
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("main.cpp", Form_main); // Головне вікно
USEFORM("about.cpp", Form_about); // Вікно даних про розробника
USEFORM("light.cpp", Form_light); // Вікно управління світлом
USEFORM("water.cpp", Form_water); // Вікно управління водою
USEFORM("gas.cpp", Form_gas); // Вікно управління газом
USEFORM("security.cpp", Form_security); // Вікно управління захистними системами
USEFORM("climate.cpp", Form_climate); // Вікно управління кліматом
USEFORM("phone.cpp", Form_phone); // Вікно управління телефонією
USEFORM("tv.cpp", Form_tv); // Вікно управління телевізорами
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                // Створення головної форми
                Application->CreateForm(__classid(TForm_gas), &Form_gas);
                // Створення форми управління захистними системами
                Application->CreateForm(__classid(TForm_security),
&Form_security);
                // Створення форми управління кліматом
                Application->CreateForm(__classid(TForm_climate),
&Form_climate);
                // Створення форми управління телефонією
                Application->CreateForm(__classid(TForm_phone), &Form_phone);
                // Створення форми управління телевізорами
                Application->CreateForm(__classid(TForm_main), &Form_main);
                // Створення форми даних про розробника
                Application->CreateForm(__classid(TForm_about), &Form_about);
                // Створення форми управління світлом
                Application->CreateForm(__classid(TForm_light), &Form_light);
                // Створення форми управління водою
                Application->CreateForm(__classid(TForm_water), &Form_water);
                // Створення форми управління газом
                Application->CreateForm(__classid(TForm_tv), &Form_tv);
                // Створення форми запуску проекту
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Файл CPU_Modbus_RTU_Socket.cpp – протокол Modbus_RTU

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

#include <basepch.h>
#pragma hdrstop
#include "CPU_Modbus_RTU_Socket.h"
#pragma package(smart_init)
//------- -------------------
//
static inline void ValidCtrCheck(TCPU_Modbus_RTU_Socket *)
{
    new TCPU_Modbus_RTU_Socket(NULL);
}
//#include "OcelotStateUnit.cpp"
//------- -------------------
__fastcall TCPU_Modbus_RTU_Socket::TCPU_Modbus_RTU_Socket(TComponent* Owner)
    : TClientSocket(Owner)
{
_Modbus_RTU_StateChangeNum=new T_Modbus_RTU_StateChangeNum;
OcelotStateChangeNum=new TOcelotStateChangeNum;
_Modbus_RTU_State=new T_Modbus_RTU_State;
OcelotState=new TOcelotState;

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

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

}

__fastcall TCPU_Modbus_RTU_Socket::~TCPU_Modbus_RTU_Socket(void)
{
delete _Modbus_RTU_StateChangeNum;
delete OcelotStateChangeNum;
delete _Modbus_RTU_State;
delete OcelotState;

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

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

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

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

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

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

NewLoop:

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

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


switch ( local_pointer[0] )
{
case  'a':
            if (CommandStrSize==2)
                {
                FAuth=local_pointer[1];
                if (FOnChangeAuthStatus)
```

```
                    FOnChangeAuthStatus(this,FAuth);
                };
            break;

case  'c': break;

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

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

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

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


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

end_label:
free(buff);
}
//------- ------------------
bool __fastcall TCPU_Modbus_RTU_Socket::SendBuf(char *buf,long size)
{
if (Active)
    {
    char *buff;
    long size_buff=size+7;
    buff=(char *)malloc(size_buff);
    memcpy(buff,"CPUXA",5);
    memcpy(&buff[5],&size,2);
```

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

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

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

```
}
//------- -------------------
void __fastcall TCPU_Modbus_RTU_Socket::GetInternalProtocolVersion(void)
{
char buf[1];
buf[0]='p';
SendBuf(buf,1);
}
//------- -------------------
void __fastcall T_Modbus_RTU_StateChangeNum::Next(void)
{
if (Num==65534)
    Num=1;
else
    Num++;
};
//------- -------------------
__fastcall T_Modbus_RTU_StateChangeNum::T_Modbus_RTU_StateChangeNum(void)
{
Num=1;
};
//------- -------------------
AnsiString __fastcall T_Modbus_RTU_State::GetStateOnOff(unsigned char house,
unsigned char key)
{
if (map[house][key]==true)
return "On";
else
return " --";
}
//------- -------------------
bool __fastcall T_Modbus_RTU_State::GetOnOffStatus(unsigned char house, unsigned
char key)
{
if (map[house][key]==true)
return true;
else
return false;
}
//------- -------------------
bool __fastcall T_Modbus_RTU_State::LoadInfoBuff(char *buff)
{
memcpy(map,buff,256);
return true;
}
//------- -------------------
void _fastcall TCPU_Modbus_RTU_Socket::SetLogin(AnsiString str)
{
int len;
len=str.Length();
if (len<5)
    {
    for(int i=len;i<5;i++)
        str=str+"1";
    }
FLogin=str.SubString(1,5);
return;
}
void _fastcall TCPU_Modbus_RTU_Socket::SetPassword(AnsiString str)
{
int len;
len=str.Length();
if (len<10)
    for(int i=len;i<10;i++)
        str=str+"1";
FPassword=str.SubString(1,10);
return;
}
```

Файл CPU_Modbus_RTU_Socket.h - бібліотека для файлу CPU_Modbus_RTU_Socket.cpp

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

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

#include "OcelotStateUnit.h"


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

class T_Modbus_RTU_StateChangeNum {
public:           // Визначено користувачем
unsigned short Num;
void __fastcall Next(void);
__fastcall T_Modbus_RTU_StateChangeNum(void);
};

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


class PACKAGE TCPU_Modbus_RTU_Socket : public TClientSocket
{
private:
    AnsiString FVersion;
    AnsiString FLogin;
    AnsiString FPassword;
    unsigned char FAuth;
    TOcelotStateChangeNum *OcelotStateChangeNum;
    T_Modbus_RTU_StateChangeNum *_Modbus_RTU_StateChangeNum;
    TOcelotState *OcelotState;

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

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

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

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

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

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

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

#endif
```

Файл main.cpp основної програми

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

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

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

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


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

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

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

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

```
//відкриття вікна "Система безпеки"
void __fastcall TForm_main::Button3Click(TObject *Sender)
{
Form_security->Show();
}
//---------------------------------------------------------------------

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

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

Файл main.h – бібліотека для файлу main.cpp

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

Файл climate.cpp – керування кліматом

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

Файл climate.h – бібліотека для файлу climate.cpp

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

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

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

Файл light.cpp – керування освітленням

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

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

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

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


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

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

ClientSocket->Send_Modbus_RTU_Command(0,2,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,2,TrackBar_corridor->Position);

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

//ввімкнення світла у вітальні з вказаною яскравістю
void __fastcall TForm_light::drawing_room_onClick(TObject *Sender)
{
```

```
drawing_room->Caption="Ввімкнено";
Image_drawing_room_on->Visible=true;
Image_drawing_room_off->Visible=false;
TrackBar_drawing_room->Enabled=true;

ClientSocket->Send_Modbus_RTU_Command(0,3,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,3,TrackBar_drawing_room->Position);
}
//---------------------------------------------------------------------------

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

ClientSocket->Send_Modbus_RTU_Command(0,4,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,4,TrackBar_kitchen->Position);
}
//---------------------------------------------------------------------------

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

ClientSocket->Send_Modbus_RTU_Command(0,5,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,5,TrackBar_cabinet->Position);
}
//---------------------------------------------------------------------------

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

ClientSocket->Send_Modbus_RTU_Command(0,6,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,6,TrackBar_bedroom->Position);
}
//---------------------------------------------------------------------------

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

ClientSocket->Send_Modbus_RTU_Command(0,7,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,7,TrackBar_baby_room->Position);
}
//---------------------------------------------------------------------------

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

```
ClientSocket->Send_Modbus_RTU_Command(0,8,18,1);
ClientSocket->SendLeviton_Modbus_RTU_(0,8,TrackBar_bathroom->Position);
}


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

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

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

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

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

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

//вимкнення світла у дитячій кімнаті
void __fastcall TForm_light::baby_room_offClick(TObject *Sender)
{
baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
```

```
TrackBar_baby_room->Enabled=false;
ClientSocket->Send_Modbus_RTU_Command(0,7,19,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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


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

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

//імітація присутності господарів
//ввімкнення та вимкнення світла випадковим чином
```

```
void __fastcall TForm_light::Button1Click(TObject *Sender)
{

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ClientSocket->Send_Modbus_RTU_StatusQuery();

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

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

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

```
TrackBar_drawing_room->Enabled=false;
}
st=_Modbus_RTU_State->GetStateOnOff(0, 4);
if(st="On") {
kitchen->Caption="Ввімкнено";
Image_kitchen_on->Visible=true;
Image_kitchen_off->Visible=false;
TrackBar_kitchen->Enabled=true;
}
else {
kitchen->Caption="Вимкнено";
Image_kitchen_off->Visible=true;
Image_kitchen_on->Visible=false;
TrackBar_kitchen->Enabled=false;
}
st=_Modbus_RTU_State->GetStateOnOff(0, 5);
if(st="On") {
cabinet->Caption="Ввімкнено";
Image_cabinet_on->Visible=true;
Image_cabinet_off->Visible=false;
TrackBar_cabinet->Enabled=true;
}
else {
cabinet->Caption="Вимкнено";
Image_cabinet_off->Visible=true;
Image_cabinet_on->Visible=false;
TrackBar_cabinet->Enabled=false;
}
st=_Modbus_RTU_State->GetStateOnOff(0, 6);
if(st="On") {
bedroom->Caption="Ввімкнено";
Image_bedroom_on->Visible=true;
Image_bedroom_off->Visible=false;
TrackBar_bedroom->Enabled=true;
}
else {
bedroom->Caption="Вимкнено";
Image_bedroom_off->Visible=true;
Image_bedroom_on->Visible=false;
TrackBar_bedroom->Enabled=false;
}
st=_Modbus_RTU_State->GetStateOnOff(0, 7);
if(st="On") {
baby_room->Caption="Ввімкнено";
Image_baby_room_on->Visible=true;
Image_baby_room_off->Visible=false;
TrackBar_baby_room->Enabled=true;
}
else {
baby_room->Caption="Вимкнено";
Image_baby_room_off->Visible=true;
Image_baby_room_on->Visible=false;
TrackBar_baby_room->Enabled=false;
}
st=_Modbus_RTU_State->GetStateOnOff(0, 8);
if(st="On") {
bathroom->Caption="Ввімкнено";
Image_bathroom_on->Visible=true;
Image_bathroom_off->Visible=false;
TrackBar_bathroom->Enabled=true;
}
else {
bathroom->Caption="Вимкнено";
Image_bathroom_off->Visible=true;
Image_bathroom_on->Visible=false;
TrackBar_bathroom->Enabled=false;
}
}
```

Файл light.h – бібліотека для файлу light.cpp

```
#ifndef lightH
#define lightH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm_light : public TForm
{
__published:      // Компоненти IDE-управління
        TTrackBar *TrackBar_corridor;
        TLabel *Label11;
        TImage *Image_torch_on;
        TImage *Image_torch_off;
        TTrackBar *TrackBar_torch;
        TLabel *Label8;
        TImage *Image_corridor_on;
        TImage *Image_corridor_off;
        TTrackBar *TrackBar_drawing_room;
        TLabel *Label9;
        TImage *Image_drawing_room_on;
        TImage *Image_drawing_room_off;
        TTrackBar *TrackBar_cabinet;
        TLabel *Label12;
        TImage *Image_cabinet_on;
        TImage *Image_cabinet_off;
        TTrackBar *TrackBar_bedroom;
        TLabel *Label13;
        TImage *Image_bedroom_on;
        TImage *Image_bedroom_off;
        TTrackBar *TrackBar_baby_room;
        TLabel *Label14;
        TImage *Image_baby_room_on;
        TImage *Image_baby_room_off;
        TTrackBar *TrackBar_kitchen;
        TLabel *Label15;
        TImage *Image_kitchen_on;
        TImage *Image_kitchen_off;
        TTrackBar *TrackBar_bathroom;
        TLabel *Label16;
        TImage *Image_bathroom_on;
        TImage *Image_bathroom_off;
        TButton *Button17;
        TButton *Button18;
        TLabel *Label17;
        TLabel *torch;
        TLabel *Label19;
        TLabel *corridor;
        TLabel *Label21;
        TLabel *drawing_room;
        TLabel *Label23;
        TLabel *kitchen;
        TLabel *Label25;
        TLabel *cabinet;
        TLabel *Label27;
        TLabel *bedroom;
        TLabel *Label29;
        TLabel *baby_room;
        TLabel *Label31;
        TLabel *bathroom;
        TBevel *Bevel1;
        TLabel *Label10;
        TBevel *Bevel2;
```

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

Файл water.cpp – керування системою водопостачання

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Файл water.h – бібліотека для файлу water.cpp

```
#ifndef waterH
#define waterH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm_water : public TForm
{
__published:      // Компоненти IDE-управління
        TGroupBox *GroupBox1;
        TButton *Button1;
        TButton *Button2;
        TLabel *Label1;
        TLabel *Label_1;
        TGroupBox *GroupBox2;
        TLabel *Label3;
        TLabel *Label_4;
        TButton *Button3;
        TButton *Button4;
        TGroupBox *GroupBox3;
        TLabel *Label5;
        TLabel *Label_2;
        TButton *Button5;
        TButton *Button6;
        TGroupBox *GroupBox4;
        TLabel *Label7;
        TLabel *Label_3;
        TButton *Button7;
        TButton *Button8;
        TGroupBox *GroupBox5;
        TLabel *Label9;
        TLabel *Label_5;
        TButton *Button9;
        TButton *Button10;
        TGroupBox *GroupBox6;
        TLabel *Label11;
        TLabel *Label_6;
        TButton *Button11;
        TButton *Button12;
        TLabel *Label13;
        TLabel *Label_11;
        TLabel *Label15;
        TLabel *Label_44;
        TTimer *Status;

        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
        void __fastcall Button12Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button9Click(TObject *Sender);
        void __fastcall Button11Click(TObject *Sender);
        void __fastcall StatusTimer(TObject *Sender);
```

```
private:    // Визначено користувачем
public:           // Визначено користувачем
        __fastcall TForm_water(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm_water *Form_water;
//---------------------------------------------------------------------------
#endif
```

Файл gas.cpp – керування системою газопостачання

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

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

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

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

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

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

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

}
```

Файл gas.h – бібліотека для файлу gas.cpp

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

Файл tv.cpp - керування дломашнім кінотеатром

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

#include <vcl.h>
#pragma hdrstop

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


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

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

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

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

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

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

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

Файл tv.h – бібліотека для файлу tv.cpp

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

Файл phone.cpp – керування телефонним зв'язком

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

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

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

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

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

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

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

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

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

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

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

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

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

Файл phone.h – бібліотека для файлу phone.cpp

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

Файл security.cpp – керування системою безпеки

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

#include "security.h"
#include "CPU_Modbus_RTU_Socket"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_security *Form_security;
//---------------------------------------------------------------------------
__fastcall TForm_security::TForm_security(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_security::Button14Click(TObject *Sender)
{
ClientSocket->Send_Modbus_RTU_Command(0,17,18,1);
}
//---------------------------------------------------------------------------

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

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

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

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

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

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

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

//визначення та виведення стану системи безпеки
```

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

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

st=_Modbus_RTU_State->GetStateOnOff(0, 19);
if(st="On") Label_3->Caption="Ввімкнено"; else Label_3->Caption="Вимкнено";
st=_Modbus_RTU_State->GetStateOnOff(0, 20);
if(st="On") Label_4->Caption="Заблоковано"; else Label_4->Caption="Відкрито";


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

Файл security.h – бібліотека для файлу security.cpp

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

Файл about.cpp – довідка про програму

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

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

void __fastcall TForm_about::FormCreate(TObject *Sender)
{
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("МАГІСТЕРСЬКИЙ ПРОЕКТ");
Memo1->Lines->Add("");
Memo1->Lines->Add("на  тему:");
Memo1->Lines->Add("");
Memo1->Lines->Add("  Дослідження та програмна реалізація системи мережного керування помешканнями з використанням протоколу Modbus/RTU");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("Керівник: Якименко Н.М.");
Memo1->Lines->Add("");
Memo1->Lines->Add("Розробив: студент Сопілка Богдан Юрійович");
Memo1->Lines->Add("                        гр. КН-22М-2");
Memo1->Lines->Add("");
Memo1->Lines->Add("");
Memo1->Lines->Add("м. Кропивницький 2023");
}
```

Файл about.h – бібліотека для файлу about.cpp

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

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