МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТАВРІЙСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.І.ВЕРНАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ МУНІЦИПАЛЬНОГО УПРАВЛІННЯ ТА МІСЬКОГО ГОСПОДАРСТВА
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

До захисту допущено

Завідувач кафедри
_____Олександр ГУЙДА

"___" _____ 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
до магістерської кваліфікаційної роботи освітнього ступеня "магістр"

з галузі знань 12 «Інформаційні технології»

спеціальності 122 «Комп'ютерні науки»

на тему: Дослідження обміну даними по комп'ютерним мережам на основі протоколів Modbus

Студент групи КНм – 61 Руденко В'ячеслав Віталійович _____
(шифр групи) (прізвище, ім'я, по батькові) (підпис)

Керівник роботи к.т.н., доцент Лісовець С.М. _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Консультанти:
охорона праці та навколишнього середовища доцент Гуйда О.Г. _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Київ – 2023

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

ТАВРІЙСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.І.ВЕРНАДСЬКОГО

Навчально-науковий інститут муніципального управління та міського господарства

Другий (магістерський) освітній рівень

Галузь знань 12 «Інформаційні технології»
(шифр і назва)

Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____Олександр ГУЙДА

“___” _____ 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту ___Руденко В’ячеславу Віталійовичу ____
(прізвище, ім’я, по батькові)

1 Тема роботи: Дослідження обміну даними по комп’ютерним мережам на основі протоколів Modbus

керівник роботи ___к.т.н., доцент Лісовець С.М.,___
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені ректором Університету від “ 25 ”вересня 2023 року

2 Строк подання студентом роботи “02” грудня 2023р.

3 Вихідні дані до роботи:

1) Науково-технічні публікації, що досліджують обмін даними по комп'ютерним мережам, зокрема застосування протоколів Modbus у сучасних системах зв'язку та автоматизації;
2) Результати науково-дослідних робіт з області інформаційних технологій, зокрема в розробці та оптимізації програмного забезпечення для ефективного обміну даними за допомогою протоколу Modbus в різноманітних сценаріях використання.

4 Зміст розрахунково-пояснювальної записки:

4.1 Проаналізувати основне визначення та роль протоколів зв'язку у контексті комп'ютерних мереж;

4.2 Розглянути загальні характеристики обміну даними, специфічні для комп'ютерних мереж.

4.3 Розглянути різновиди архітектур мереж, в яких може використовуватися Modbus;

4.4 Визначити загрози безпеки та запропоновано заходи захисту в системах, які використовують Modbus;

5 Перелік графічного матеріалу:

Графічна робота виконана у вигляді мультимедійної презентації

6 Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Загальна частина	доцент Лісовець С.М.		
Технологічна частина	доцент Лісовець С.М.		
Спеціальна частина	доцент Лісовець С.М.		
Охорона праці та навколишнього середовища	доцент Гуйда О.Г.		
Екномічна частина	доцент Лісовець С.М.		
Графічна частина	ст. викладач Фуртат О.В.		

7 Дата видачі завдання «26» вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Загальна частина	*жовтень*	
Технологічна частина	*листопад*	
Спеціальна частина	*листопад*	
Охорона праці та навколишнього середовища	*жовтень*	
Економічна частина	*листопад*	
Графічна частина	*грудень*	

Студент _____ В’ячеслав РУДЕНКО
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сергій ЛІСОВЕЦЬ
(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ……05

ВСТУП……6

1 ОГЛЯД ПРОТОКОЛІВ MODBUS……9

1.1 Вступ до протоколів зв'язку……9

1.2 MODBUS як протокол зв'язку……12

2 МЕХАНІЗМИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ MODBUS……16

2.1 Огляд системи……16

2.2 Реалізація протоколу на мережевому рівні……18

3 ЗАСТОСУВАННЯ MODBUS В РЕАЛЬНИХ СИСТЕМАХ……24

3.1 Промислові застосування протоколу……24

3.2 Проблеми та виклики використання MODBUS……63

3.3 Перспективи розвитку MODBUS……66

4 ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬОГО СЕРЕДОВИЩА……68

5 ЕКОНОМІЧНА ЧАСТИНА……71

ЗАГАЛЬНИЙ ВИСНОВОК……72

ПЕРЕЛІК ПОСИЛАНЬ……74

РЕФЕРАТ

У магістерській кваліфікаційній роботі досліджено протоколи зв'язку, зокрема Modbus, з урахуванням їх застосування в промислових системах автоматизації. Робота включає 100 сторінок, 25 ілюстрацій, 10 таблиць, 5 креслень та 3 додатки. Бібліографічний список містить 50 найменувань.

Загальна характеристика та актуальність теми розкриті з урахуванням стрімкого росту використання Modbus у промисловості. Мета роботи - дослідити принципи функціонування та реалізації протоколу, визначити проблеми й вирішення на різних рівнях його застосування.

У роботі висвітлено нові аспекти використання Modbus у системах автоматизації, а також розроблено технічні рішення для забезпечення безпеки обміну даними. Отримані результати включають нові моделі систем автоматизації, які демонструють покращену ефективність та безпеку.

Робота була представлена на п'яти міжнародних конференціях, що підтверджується п'ятьма статтями у фахових виданнях.

Зміст роботи охоплює огляд протоколів Modbus, механізми та засоби їх реалізації, а також застосування Modbus в реальних системах. Висновки роботи надають загальні висновки щодо використання та рекомендації щодо впровадження розроблених досліджень.

ВСТУП

Мережі IoT стають все більш популярними з розвитком технологій. Взаємозв'язані пристрої вдома, офісі чи промислових будівлях досить поширені для моніторингу та контролю навколишнього середовища. Сьогодні легко отримати голосового помічника, підключитися до домашньої мережі ethernet, керувати світлом і звуком в будинку.

Продукція на ринку поширена і відома. Однак споживачі можуть бути стурбовані конфіденційністю та безпекою домогосподарства та приватного життя, коли всі пристрої знаходяться в одній мережі. Пристрої IoT можуть бути вразливими для хакерів, оскільки користувачі можуть бути необережними або на певному пристрої існують помилки, які дозволяють несанкціонований доступ до особистих даних, наприклад історії голосових команд.

Протокол Modbus RTU є чудовою альтернативою бездротовим пристроям Інтернету речей, підключеним до мережі Ethernet. Мережа Modbus RTU на основі послідовного з'єднання RS-485 складається з головного пристрою та до 247 призначених для користувача пристроїв, які підтримують

Modbus RTU або сертифіковані Modbus RTU пристрої на великі відстані.

Протокол Modbus сам по собі є єдиним інструментом для взаємодії між пристроями. Оскільки протокол Modbus є основним міжмашинним протоколом, окреме програмне забезпечення має обробляти мережу Modbus і взаємодіяти з хмарою IoT-Ticket.

У цьому налаштуванні персональний комп'ютер із операційною системою Ubuntu служить головним пристроєм у мережі Modbus. Для створення використовувався пристрій USB to RS-485 програмне забезпечення, яке буде прошарком між мережею Modbus і сховищем даних, у цьому випадку IoT-TICKET обрано як платформу для зберігання та керування даними IoT.

Програмне забезпечення, яке дозволяє пристроям Modbus і хмарі IoT-TICKET працювати як одна система, було запрограмовано за допомогою мови програмування Python завдяки його високій універсальності та багатьом бібліотекам.

Однак, оскільки операційна система, наприклад Microsoft Windows, також підтримує Python, можна використовувати домашній сервер як головний пристрій Modbus або перетворити Raspberry Pi на головний пристрій Modbus.

Система, розроблена в рамках цієї дисертації, також може бути застосована до автоматизації та досліджень проектування пристроїв загалом. Будучи відкритим протоколом, ця система може дозволити студентам проектувати, розробляти та впроваджувати власні пристрої, взаємодіючи з системою, дозволяючи їм застосовувати теоретичні знання на практиці.

Анотація роботи: Пояснювальна записка магістерської кваліфікаційної роботи складається із 100 сторінок, включає 25 ілюстрацій, 10 таблиць, 5 креслень та 3 додатків. Бібліографічний список містить 50 найменувань. **Загальна характеристика та актуальність теми:** Робота присвячена вивченню та аналізу сучасних тенденцій у розвитку протоколів зв'язку, зокрема Modbus. З урахуванням стрімкого росту використання цього протоколу в промисловості, актуальність дослідження полягає в необхідності забезпечення його ефективності та безпеки.

Мета дослідження та визначення задач: Мета роботи - дослідити принципи функціонування та реалізації протоколу Modbus, визначити проблеми й вирішення на різних рівнях його застосування.

Наукова новизна роботи: В роботі розкрито нові аспекти використання Modbus в системах автоматизації та запропоновано технічні рішення для забезпечення безпеки обміну даними.

Отримані результати: В роботі розроблені та реалізовані технічні рішення для оптимізації передачі даних за допомогою Modbus. Отримані результати включають нові моделі систем автоматизації, які демонструють покращену ефективність та безпеку.

Апробація роботи:

Результати роботи доповідалися на міжнародних конференціях, що підтверджується п'ятьма статтями у фахових виданнях. А саме:

Результати дослідження "Дослідження обміну даними по комп'ютерним мережам на основі протоколів Modbus" були представлені та обговорені на ряді наукових заходів, підтверджуючи актуальність та значущість отриманих висновків.

Апробація роботи включає в себе наступні етапи:

1. Міжнародна конференція з інформаційних технологій (ICT'2023):

- Доповідь на тему "Роль протоколів Modbus у сучасних комп'ютерних мережах" на пленарному засіданні.
- Представлення тез доповіді та участь у панельній дискусії.

2. Науковий семінар "Мережеві технології та інформаційна безпека (NetSec'2023):

- Доповідь "Архітектура та основні характеристики протоколу Modbus" на секції з мережевих технологій.
- Обговорення результатів та взаємодія з іншими учасниками семінару.

3. Участь у воркшопі "Сучасні технології зв'язку та їх вплив на промисловість (CommTech'2023):

- Презентація розроблених технічних рішень для оптимізації обміну даними за допомогою протоколів Modbus.
- Залучення учасників до обговорення практичних аспектів впровадження вироблених систем.

4. Публікація статті в науковому журналі "Інформаційні технології та мережі (ITN'2023)":

- Розміщення докладних результатів дослідження у визначеному науковому виданні.
- Запрошення наукової спільноти до обговорення та подальших досліджень у цій області.

5. Участь у вебінарі з теми "Сучасні тенденції у використанні протоколів зв'язку (WebComm'2023)":

- Виступ зі зведеними результатами та акцентом на практичні застосування.
- Відповіді на запитання учасників вебінару та взаємодія з аудиторією.

Апробація роботи на цих заходах сприяла обговоренню, вдосконаленню та підтвердженню цінності отриманих результатів у галузі дослідження обміну даними по комп'ютерним мережам на основі протоколів Modbus.

1 ОГЛЯД ПРОТОКОЛІВ MODBUS

1.1 Вступ до протоколів зв'язку

У сучасному підключеному світі зв’язок між датчиками та системами є ключовим. Простий і надійний спосіб автоматичного збору виміряних даних є обов’язковим для безперебійної роботи багатьох процесів. Увімкнення легкого збору даних економить час і гроші, оскільки технічне обслуговування системи спрощується завдяки централізованому головному блоку, який контролює кожен датчик. При правильному застосуванні система може заощадити ресурси компанії та підвищити ефективність.

Завдяки широкій підтримці протоколу Modbus [1] для зв’язку між промисловими пристроями та датчиками можливість його підтримки є величезною перевагою в пропозиції конкурентоспроможного промислового застосування.

Вбудована система — це комп’ютерна система, призначена для попереднього виконання конкретного завдання. Вбудовані системи є невеликими за розміром, малопотужними та дешевими системами. Розробка системи для виконання лише невеликого набору завдань гарантує, що жодні ресурси не будуть витрачені даремно чи зайві.

Розробляти вбудовані системи з низьким енергоспоживанням важко. Термін «низька потужність» передбачає різні властивості системи залежно від конкретної програми. Системи, які повинні працювати безперервно, повинні зосередитися на зниженні своєї динамічної потужності, щоб бути ефективними. Пристрої, що живляться від акумулятора, повинні зосередитися набагато більше на оптимізації використання режимів сну[6].

Мікропроцесори зазвичай використовуються для додавання інтелекту до вбудованої системи. Це дешевий і простий спосіб додати логіку та функціональність системі, чого інакше було б важко досягти.

Архітектура

Мікропроцесори бувають різноманітних форм і розмірів. Їх базова структура може значно відрізнятися залежно від виробника, архітектури та набору інструкцій.

Сімейство мікропроцесорів Cortex охоплює широкий спектр випадків використання. Від потужніших процесорів на базі Cortex, M3-M4 і M7, до менших і більш енергоефективних ядер M0 і M0+. Основною відмінністю між M0 і M0+ є менший двоступеневий конвеєр M0+. Обидва вони базуються на одній архітектурі ARMv6-M і підтримують однаковий набір інструкцій.

Як приклад [7] мікропроцесора M0+ EFM32ZG108 [8] Silicon Lab буде використано для опису функціональності та особливостей мікропроцесора.

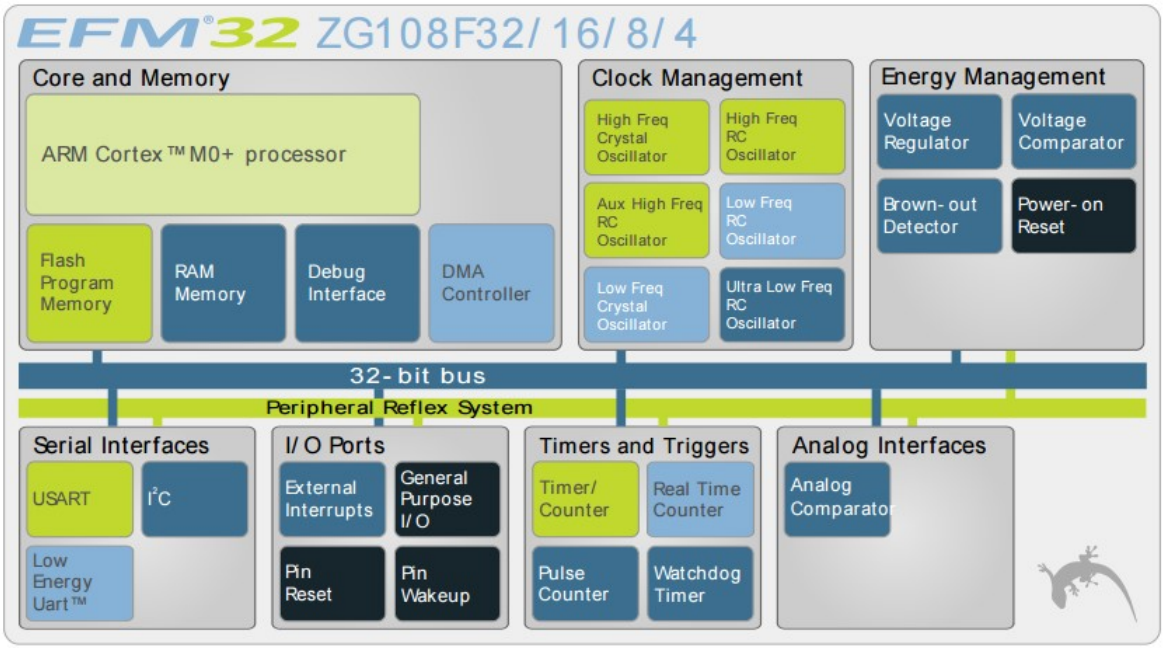


Рисунок 1.1- Блок-схема EFM32ZG108F32 [8].

На рисунку 1.1 показано блок-схему внутрішньої структури EFM32ZG108F32. У верхній половині діаграми зображені основні частини мікропроцесора, а саме ядро, пам'ять, тактова частота та блок живлення.

Пам'ять

Як видно у верхньому лівому куті рисунка 2.1, процесор має два типи внутрішньої пам'яті. EFM32ZG108F32 постачається з 32 КБ флеш-пам’яті та 4 КБ оперативної пам’яті, RAM.

Процесор також підтримує так званий Direct Memory Access, DMA. DMA — це функція, яка дозволяє певним периферійним пристроям читати/записувати з пам’яті, не займаючи ядра. Це дозволяє ядру виконувати інші завдання або переходити в сплячий режим під час передачі даних.

Периферійні пристрої

Мікропроцесор зазвичай має деякі вбудовані функції, які зазвичай використовуються, наприклад GPIO, таймери, UART, I2C, SPI, DMA, щоб назвати декілька. Ці периферійні пристрої підтримуються в апаратному забезпеченні на тому ж чіпі, що й сам логічний блок, і можуть використовуватися, якщо цього вимагає програма, без додавання додаткового обладнання.

Конфігурація годинника

Блок керування годинником, CMU, складається з конфігурованого дерева годинника. EFM32ZG108F32 має два внутрішні кристали, один низькочастотний тактовий сигнал, що працює на 32 кГц, і один високочастотний тактовий сигнал, що працює на 21 МГц [8]. Також можна підключити два зовнішні годинники, один низькочастотний і один високочастотний для кращої точності та різних базових тактових частот.

Синхронізація високої або низької частоти може використовуватися для тактування кожного периферійного пристрою залежно від того, як воно передбачається використовувати в програмі.

I/O

Основне призначення вбудованої системи – обробка даних. Щоб це було можливим, мають бути дані для обробки. Вбудована система, швидше за все, буде підключена до якогось датчика або приводу, щоб служити певній меті.

Зовнішні інтерфейси

Якщо мікропроцесор не підтримує необхідний інтерфейс або функціональність, до системи можна додати зовнішню мікросхему. Маленькі екрани зазвичай додають для забезпечення візуального зворотного зв’язку під час роботи системи.

2.1.4 Споживання електроенергії

Вбудовані системи, як правило, розраховані на мінімальну потужність. Для досягнення цього мікропроцесори зазвичай мають певну функцію режиму сну. У режимі сну мікропроцесор вимикає певні функції, як-от живлення ядра процесора, годинник і периферійні пристрої, щоб зберегти енергію.

1.2 MODBUS як протокол зв'язку

Протокол Modbus є найбільш поширеним промисловим протоколом, призначеним для взаємодії між машинами; Розроблений компанією Modicon, пізніше придбаною Schneider Electric, протокол був представлений на ринку в 1979 році. Schneider Electric допомогла в розробці незалежних організацій спільноти розробників і користувачів.

Відкритий вихідний код і універсальність протоколу дозволяють багатьом виробникам прийняти протокол і впровадити його у своє виробництво. Основними сферами застосування протоколу Modbus є керування реле та контролерами, збір даних з датчиків та моніторинг.

Система, реалізована в цій роботі, зосереджена на форматі кадру MODBUS RTU на напівдуплексній лінії передачі даних RS-485.

Формат кадру Modbus RTU кодує дані у двійковий формат; часовий інтервал виконує роль розділювача пакетів. Цей протокол має низьку толерантність до затримки.

Пристрої Modbus взаємодіють за моделлю головний-підлеглий. Всі запити виконуються тільки головним пристроєм; Підлеглі пристрої можуть лише відповідати на запити і не можуть самостійно почати передачу даних.

Мережа Modbus може складатися з кількох сегментів; однак один сегмент може включати лише один головний пристрій і 247 підлеглих пристроїв. Кожен підлеглий пристрій має унікальну адресу в мережі в діапазоні від 1 до 247. Головний пристрій не має адреси. Однак головний пристрій може надіслати пакетну адресу з адресою 0, яка транслюватиме пакет на всі підлеглі пристрої; підлеглі пристрої не можуть відповідати на широкомовні запити. Трансляція не буде реалізована в рамках цієї дипломної роботи, оскільки немає відповіді підтвердження від підлеглих пристроїв.

Модуль прикладних даних MODBUS RTU складається з:

1. Адреса підлеглого пристрою – адреса пристрою одержувача пакету в діапазоні від 0 до 247. (0 для широкомовних розсилок)

2. Блок даних протоколу — Основна частина пакету, яка включає код функції та дані. Дані залежать від коду функції — максимальний розмір 253 байти.

3. Контрольна сума — MODBUS RTU використовує алгоритм CRC16.

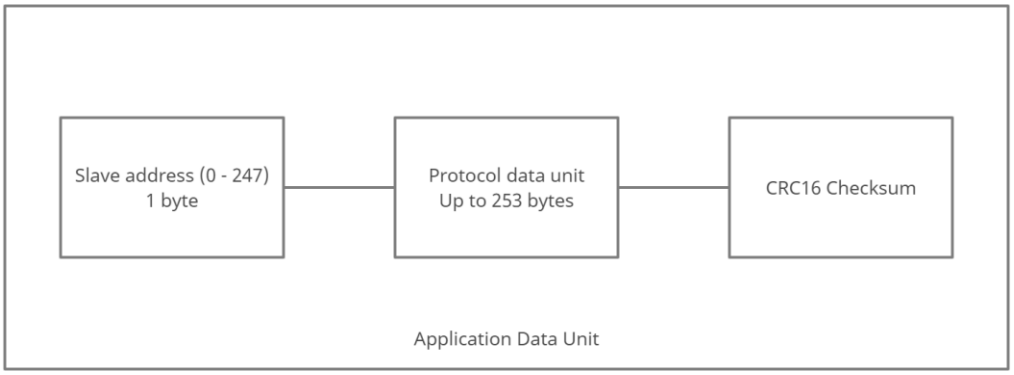


Рисунок 1.2 Представлення ADU протоколу MODBUS

4 Регістри та функції MODBUS

4.1 Типи регістрів

Підлеглий пристрій MODBUS містить чотири накопичувачі даних; кожен може містити до 9999 змінних, також відомих як регістри. Використовуючи регістри, головний пристрій може надсилати дані на підлеглий пристрій або зчитувати дані підлеглого пристрою. Основним призначенням регістрів є контроль і збір даних із певного пристрою в мережі MODBUS.

1. Дискретні вихідні котушки (DO) – 1-розрядні регістри можуть бути записані або зчитані головним пристроєм. Номер реєстру визначається таким чином: починається з 0, за яким слідують чотири цифри, що представляють місцезнаходження об’єкта. Якщо головний пристрій використовує нотацію розширеного регістру, вказується п’ять цифр. Рекомендації Modbus визначають 1 для даної котушки як стан ON, і ті самі рекомендації визначають значення 0 для даної вихідної котушки як стан OFF.

2. Дискретні вхідні контакти (DI) – 1-розрядні регістри лише для читання. Номер реєстру визначається таким чином: починається з першої цифри, за якою слідують чотири цифри, що представляють місцезнаходження суб’єкта. Якщо головний пристрій використовує нотацію розширеного регістру, вказується п’ять цифр. Рекомендації Modbus визначають 1 для даної котушки як стан ON, і ті самі рекомендації визначають значення 0 для даної вихідної котушки як стан OFF.

3. Аналоговий вхідний регістр (AI) – 16-розрядні регістри лише для читання. Номер реєстру визначається таким чином: починається з третьої цифри, за якою слідують чотири цифри, що представляють місцезнаходження суб'єкта. Якщо головний пристрій використовує нотацію розширеного регістру, вказується п'ять цифр.

4. Аналоговий вихідний регістр (AO) – 16-розрядні регістри можуть бути записані або зчитані головним пристроєм. Номер реєстру визначається таким чином: починається з четвертої цифри, за якою слідують чотири цифри, що представляють місцезнаходження суб'єкта. Якщо головний пристрій використовує нотацію розширеного регістру, вказується п'ять цифр.

Список функцій, наведений нижче, не є повним списком функцій, визначених прикладним протоколом Modbus, він включає лише приклади та функції, які використовуються для реалізації мети, визначеної в описі дипломної роботи.

код функції (1 байт), початкова адреса (2 байти), кількість дискретних входів (2 байти).

o PDU відповіді складається з коду функції 0x02 (1 байт), кількості байтів N* (1 байт), значень стану вхідних даних (n байтів, де n = N або N+1)

• Читати реєстри зберігання – код функції 03 використовується для читання від 1 до 125 блоків інфекційних реєстрів у віддаленому пристрої. PDU складається з коду функції (1 байт), початкової адреси (2 байти), кількості регістрів (2 байти).

o PDU відповіді складається з коду функції 0x03 (1 байт), кількості байтів N* (1 байт), регістрових значень (N* байтів)

• Читання вхідних регістрів - код функції 04 використовується для читання від 1 до 125 заразних вхідних регістрів у віддаленому пристрої. PDU складається з коду функції (1 байт), початкової адреси (2 байти), кількості регістрів (2 байти).

o PDU відповіді складається з коду функції 0x04 (1 байт), кількості байтів N* (1 байт), значень вхідного регістру (N* байтів)

• Запис однієї котушки – код функції 05 використовується для запису стану окремого виходу у стан ON або OFF у віддаленому пристрої. PDU складається з функціонального коду (1 байт), адреси підлеглого пристрою (2 байти), щоб

надіслати запит на перемикання котушки в стан головного пристрою, вказує значення 0x0000 або 0xFF00 для запиту підлеглого пристрою на перемикання котушки у стан ВИМКНЕНО (2 байти).

o PDU відповіді складається з коду функції 0x04 (1 байт), адреси підлеглого пристрою (2 байти)

• Записати один регістр – код функції 06 записує один регістр зберігання у віддалений пристрій. PDU складається з коду функції (1 байт), адреси підлеглого пристрою (2 байти), значення в діапазоні від 0x0000 до 0xFFFF (2 байти)

o Відповідь PDU складається з функціонального коду 0x06 (1 байт), реєстру Адреса (2 байти), значення реєстру (2 байти).

Протокол Modbus є широко використовуваним стандартом для взаємодії промислових машин. Розроблений компанією Modicon і пізніше підтриманий Schneider Electric, він був представлений на ринку в 1979 році. Відкритий вихідний код і універсальність протоколу сприяють його інтеграції в різноманітні системи виробництва.

Цей протокол застосовується для керування реле та контролерами, а також для збору даних з датчиків і моніторингу. У дипломній роботі розглядається система, що використовує формат кадру Modbus RTU на напівдуплексній лінії передачі даних RS-485.

Особливості протоколу включають в себе низьку толерантність до затримки, взаємодію за моделлю головний-підлеглий та обмеження на кількість підлеглих пристроїв у мережі. Також розглядаються типи регістрів, такі як дискретні вихідні котушки, дискретні вхідні контакти, аналоговий вхідний та вихідний регістри.

2 МЕХАНІЗМИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ MODBUS

2.1 Огляд системи

Мережа Modbus складається з одного головного модуля Modbus і довільної кількості підлеглих пристроїв, підключених через RS485. Підтримуються 32 slave без використання повторювачів. Протокол Modbus теоретично підтримує до 256 підлеглих пристроїв завдяки 8-бітовому полю адреси, якщо це підтримується апаратним забезпеченням.

На рисунку 2.1 показано загальне зображення повної мережі Modbus. Ведучий Modbus з'єднаний із підлеглими за шлейфовою конфігурацією.

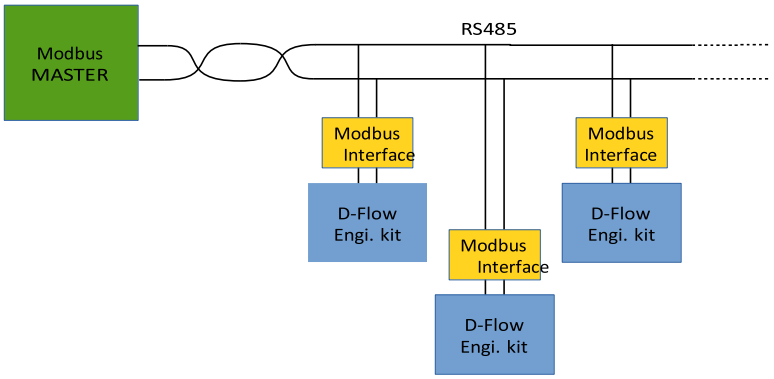


Рисунок 2.1- Огляд повної мережі Modbus

Обсяг цієї дисертації охоплює розробку та проектування підлеглого інтерфейсу Modbus між головним модулем Modbus та інженерним комплектом D-Flow. Підлеглий інтерфейс повинен мати можливість:

- Правильно підтримувати протокол Modbus
- Зчитайте дані вимірювання з інженерного комплекту
- Щоб увімкнути роботу від батареї, використовуйте мінімальну потужність
- Бути невеликого розміру

D-Flow пропонує готове до використання рішення з інженерного набору, яке реалізує їх ASIC UFO 2 для вимірювання потоку. Щоб спростити розробку системи, інженерний набір можна використовувати для швидкого створення рішень для підтвердження концептуальних рішень. На малюнку 2.2a показано зображення комплекту інженерних засобів, укладеного в стандартний корпус витратоміра. На малюнку 2.2b показано зображення UFO 2 ASIC крупним планом.

ASIC UFO2 від D-Flow — це однокристальне рішення з аналоговою вимірювальною частиною та цифровою частиною мікропроцесора. ASIC здатний вимірювати потік, температуру, потік енергії, рівень, швидкість звуку та відстань [18] [16]. Завдяки низькій потужності він здатний виконувати основні вимірювання потоку, споживаючи лише 20 мкА.

Інженерний набір підключається до ультразвукових витратомірів через серію гвинтових клем на його боці. Після цього мікропроцесор, вбудований у ASIC, налаштовується за допомогою серії послідовних команд для виконання бажаних вимірювань. UFO 2 підтримує вісім різних типів вимірювань різної інформації та довжини даних.

Потім виміряні дані надсилаються через послідовну лінію для збору в якомусь головному блоці.

2.2 Реалізація протоколу на мережевому рівні

Modbus — це гнучкий протокол у розумінні його найпростішої реалізації. У таблиці 2.1 наведено зведення необхідних властивостей для базової та звичайної реалізації разом із параметрами за замовчуванням.

Варто відзначити той факт, що Modbus не змушує вас реалізувати всі вказані функціональні коди, але дозволяє вибрати те, що потрібно для конкретної програми.

Таблиця 2.1

Карта регістрової пам'яті для типів регістрів Modbus [1].

	BASIC		REGULAR	Default value
Addressing	Slave: configurable address from 1 to	Master: to be able to address a slave from address 1 to 247	Same as Basic	-
Broadcast	Yes		Yes	-
Baud Rate	9600 (19200 is also recommended)		9600, 19200 + additional configurable baud rates	19200 (if implemented, else 9600)
Parity	EVEN		EVEN + possibility to configure NO and ODD parity	EVEN
Mode	RTU		RTU + ASCII	RTU
Electrical Interface	RS485 2W-cabling or RS232		RS485 2W-cabling (and 4W-cabling as an additional option)or RS232	RS485 2W-cabling
Connector Type	RJ 45 (recommended)			

Стек даних Modbus базується на чотирьох різних типах даних. Дискретні входи та котушки є однорозрядними значеннями даних, тоді як вхідні регістри та регістри зберігання є 16-бітними. Дискретні входи та вхідні регістри доступні лише

для читання з точки зору ведучого і діють як буфери даних для сенсорного введення в ведений пристрій. Котушки та регістри зберігання можуть бути записані головним і можуть бути налаштовані на зміну змінних і налаштувань підлеглого, якщо це необхідно. Таблиця 2.2 показує короткий перелік різних типів регістрів, оголошених специфікацією Modbus.

Modbus має певний стек пам’яті реєстрів для кожного з чотирьох типів даних, перелічених вище. Кожна з них має 10 000 визначених адрес. Таблиця 2.1 показує карту пам'яті регістра для кожного з чотирьох типів даних.

Залежно від реалізації код функції іноді діє як покажчик адреси на тип, з яким працює функція. Наприклад, під час запиту читання одного вхідного регістру поле адреси дорівнює лише 0, а не 30000.

Таблиця 2.2

Таблиця повинна мати властивості протоколу Modbus [1].

Primary tables	Object type	Type of	Comment
Discrete Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system.
Holding Registers	16-bit word	Read-Write	This type of data can be altered by an application program.

Таким чином, протокол Modbus відзначається високою розповсюдженістю, стабільністю та можливістю інтеграції в різноманітні промислові системи.

Протокол Modbus базується на функціональних кодах, які надсилаються між головним і підлеглим. Функціональні коди визначають, яку дію має бути виконано. У таблиці 2.4 показано повний набір кодів функцій, визначених протоколом Modbus.

Таблиця 2.3

Чотири основні таблиці Modbus базує свою модель даних на [1].

Register type	Address range
Discrete Output Coils	1-9999
Discrete Inputs	10000-19999
Input Registers	30000-39999
Holding Registers	40000-49999

Таблиця 2.4 Визначення коду публічної функції Modubs [1].

Function Codes						
				code	sub code	hex
Data Access	Bit Access	Physical	Read *Discrete Inputs*	2		2
		Internal bits Or	Read *coils*	1		1
			Write Single Coil	5		5
			Write Multiple *Coils*	15		0F
	16 bits access	Registers Internal Registers Or Physical	Read Input Register	4		4
			Read *Holding Registers*	3		3
			Write Single Registers	6		6
			Write Multiple Registers	16		10
			Read/Write Multiple Registers	23		17
			Mask Write Register	22		16
			Read FIFO queue	24		18
	File record access		Read File record	20		14
			Write File record	21		15
Diagnostics			Read Exception status	7		7
			Diagnostic	8	00-18,20	8
			Get Com event counter	11		0B
			Get Com event Log	12		0C
			Report Server ID	17		11
			Read device identification	43	14	2B
Other			Encapsulated Interface Transport	43	13,14	2B
			CANopen General Reference	43	13	2B

Як згадувалося раніше, не є обов'язковим впровадження всіх кодів функцій. Для невеликих додатків зазвичай реалізують лише кілька функцій читання/запису.

У таблиці 2.5 показано приклад запиту від головного модуля Modbus. Ведучий просить прочитати два вхідних регістри, 0 і 1, від підлеглого пристрою 1. Коли підлеглий пристрій отримує запит, він спочатку перевіряє, чи поле ідентифікатора підлеглого збігається з ідентифікатором підлеглого; після цього він перевіряє CRC, чи функціональний код дійсний і, нарешті, чи початкова адреса та діапазон адрес знаходяться в межах. Якщо запит вважається виконаним, підлеглий пристрій отримує запитані регістри та відповідає головному.

Код функції діє як вказівник адреси на правильний діапазон пам'яті, тому поле початкової адреси нумерується від 0-9999 замість фактичних 30000-39999.

Таблиця 2.5

Запит Modbus Master на читання двох вхідних регістрів з підлеглого.

Slave address	Function code	Starting address	Nr of registers	CRC
01	04	00 00	00 02	6B C8

Таблиця 2.6

Відповідь веденого пристрою Modbus із вмістом двох вхідних регістрів.

Slave address	Function code	Byte count	Register one	Register two	CRC
01	04	04	00 02	00 03	1A 45

Щоб майстер міг контролювати роботу мережі, використовуються коди винятків. Якщо підлеглий пристрій отримує недійсний запит, він відповідає винятковою відповіддю. Таким чином провідний (і користувач) може знати про статус підлеглих пристроїв. Типовими винятками є; неправильний функціональний код і незаконна адреса даних. Якщо головний пристрій намагається прочитати регістр, не відображений у підпорядкованій карті пам'яті Modbus, він створить виняток. У таблиці 2.7 наведено повний список вказаних кодів винятків.

У таблиці 2.8 наведено приклад головного запиту Modbus для читання 16 регістрів зберігання з підлеглого пристрою 1. Підлеглий пристрій 1 не має 16 регістрів зберігання, відображених у його карті пам'яті Modbus, тому створюється

виняток. Таблиця 2.9 показує належну відповідь на виняток від підлеглого пристрою.

Підлеглий пристрій відповідає своєю адресою, як зазвичай, але код функції має старший біт, встановлений на високий рівень, щоб вказати, що було створено виняткову ситуацію. Код винятку 02 вказує на те, що один або більше запитаних регістрів не знаходяться в межах визначеного адресного простору підлеглого пристрою.

Таблиця 2.7

Таблиця кодів винятків Modbus [1].

Code	Name
01	Illegal function
02	Illegal data address
03	Illegal data value
04	Server device failure
05	Acknowledgement
06	Server device busy
08	Memory parity error
0A	Gateway path unavailable
0B	Gateway target device failed to respond

Таблиця 2.8

Запит Modbus Master на читання 16 вхідних регістрів з підлеглого.

Slave address	Function code	Starting address	Nr of registers	CRC
01	04	00 00	00 10	F1 C6

Таблиця 2.9

Відповідь на виняткову ситуацію Modbus slave для недопустимого діапазону адрес.

Slave address	Function code	Exception code	CRC
01	84	02	C2 C1

Отже, Modbus є широко розповсюдженим, стабільним та легко інтегрується в різноманітні промислові системи.

Висока розповсюдженість та можливість вибору необхідного функціоналу роблять його ефективним для використання в різних застосунках.

3 ЗАСТОСУВАННЯ MODBUS В РЕАЛЬНИХ СИСТЕМАХ

3.1 Промислові застосування протоколу

Встановлена раніше прошивка мала свої обмеження та помилки в реалізації. На рисунку 3.1 показано послідовність дій, на основі якої була розроблена Консоль раніше. Консоль була налаштована як клієнт і мала передати пакет читання MODBUS. Він повинен був продовжувати передачу прочитаного пакету, доки не отримає жодної відповіді. Отримавши відповідь, він мав обробити дані та зберегти їх, доки не отримає весь пакет. Після цього він мав перевірити CRC і, якщо він був правильним, консоль мала вказати, що отримано правильний пакет. Також потрібно було налаштувати тайм-аут отримання, який би встановив ліміт часу для отримання пакета. Коли час очікування минув, але не було отримано весь пакет, або коли CRC був неправильним, прочитаний пакет потрібно було надіслати вдруге. Коли тайм-аут стався, коли пристрій ще отримував відповідь вдруге, пакет потрібно було скинути.

Консоль мала помилки в дизайні та реалізації в попередній роботі. Консоль продовжувала передавати пакет MODBUS, доки не отримала відповідь на надісланий раніше пакет. Це застрягло б у циклі while у програмному забезпеченні. Крім того, консоль ніколи не знала, коли настав кінець пакета. Помилки в реалізації включали нездатність консолі передати пакет запису MODBUS. Крім того, консоль змогла отримати лише чотири байти даних, а не весь пакет за раз. У Консолі виникли проблеми з впровадженням CRC для будь-яких даних. Консоль також скидалася через деякий випадковий проміжок часу, що, ймовірно, було пов’язано з тим, що не було встановлено таймер для отримання даних від блоку ретрансляції. Якщо таймер отримання не реалізовано, консоль залишатиметься в стані отримання, доки не буде виявлено будь-які дані. Це завадило консолі надсилати подальші пакети даних на будь-який інший пристрій, доки він не отримає відповідь від блоку ретрансляції. Реалізація не включала пакети помилок у свій дизайн пакетів, а також не було виконано відображення пам’яті консолі та структури танка.

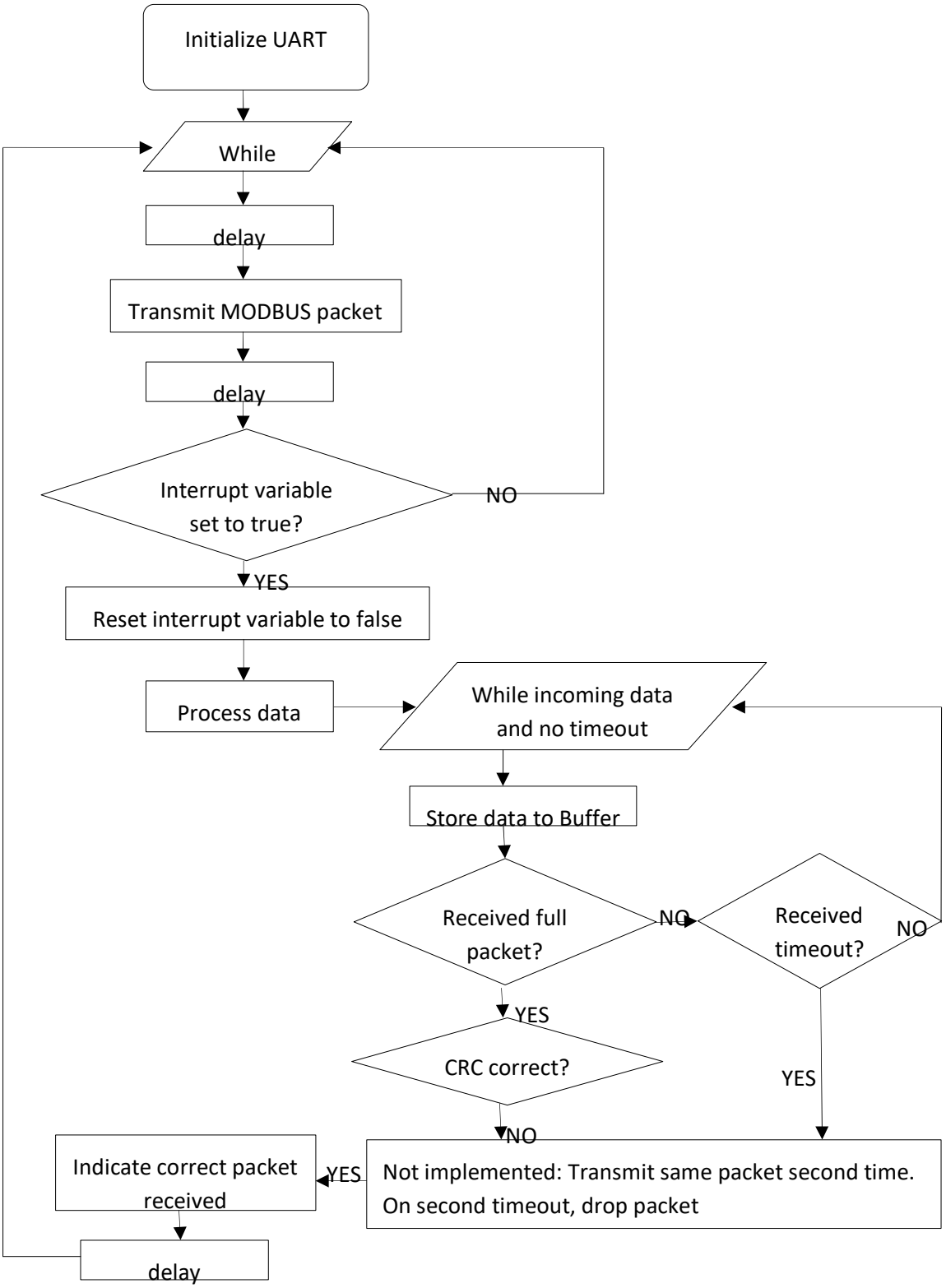


Рисунок 3.1- Попередній процес консолі

На рисунку 3.2 показано послідовність дій, на основі якої був розроблений релейний блок раніше. Блок ретрансляції був налаштований як головний і мав отримати пакет MODBUS для читання з консолі. Коли була встановлена змінна переривання прийому, що вказує на виявлення даних, потрібно було запустити процес отримання. Вхідні дані зберігалися в буфері, доки не було отримано весь

пакет. Після повного отримання пакета необхідно було перевірити CRC. Якщо CRC був правильним, пакет відповіді MODBUS мав бути переданий на консоль, а блок ретрансляції мав повернутися до стану, в якому він очікував на отримання наступних повідомлень. Якщо CRC був неправильним, потрібно було надіслати пакет помилки. Треба було налаштувати тайм-аут отримання, який би встановив ліміт часу для отримання пакета. Коли тайм-аут отримання було встановлено без отримання всього пакета, пакет потрібно було відкинути та надіслати пакет із помилкою.

Встановлене раніше мікропрограмне забезпечення було вивчено для отримання огляду дизайну та реалізації. Здавалося, що у реалізації було занадто багато помилок, а також потрібно було вирішити кілька проблем дизайну. Весь дизайн і логіку було перероблено на основі вивчення помилок попередньої роботи. Програмне забезпечення плат виконано на C/C++ [6], [7]. Для операцій компіляції та налагодження коду використовувався інструмент IAR Embedded Workbench [8].

Апаратне забезпечення консолі та релейного блоку включало процесори AVR Atmega-163 разом із 512 байтами пам’яті [9]. Для програмування плат використовувався новий програматор AVR ISP mkII. Цей новий програматор не був сумісний із платами, його апаратне забезпечення було виправлене та запрограмоване для належного функціонування [10]. Для програмування плат за допомогою програматора використовувалася програма AVR Studio [11]. Щоб досягти кінцевої мети передачі пакетів MODBUS для читання та запису та обміну відповідями між консоллю, блоком ретрансляції та персональним комп’ютером, було використано покроковий підхід.

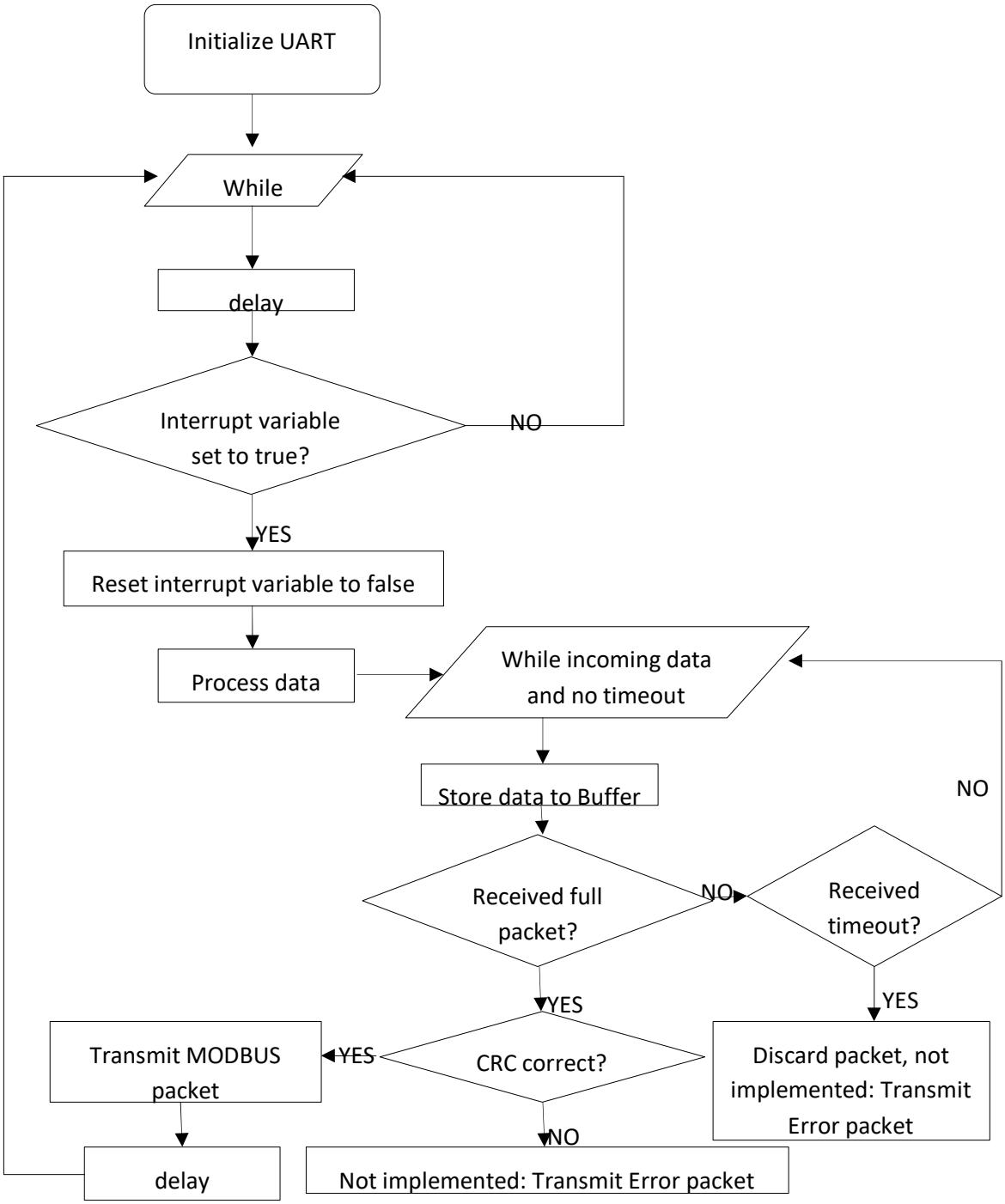


Рисунок 3.2- Послідовність дій попереднього блоку реле

Блок реле мав помилки в проектуванні та реалізації в попередній роботі. Він отримав би пакет читання MODBUS, але не зміг надіслати пакет відповіді. Він не перевірявся на отримання пакета запису MODBUS від консолі. Крім того, виникли проблеми з тим, що таймер прийому не працював належним чином. Пакети помилок не були реалізовані, а також не було реалізовано відображення пам’яті. Процедура та результати експерименту:

Ці кроки включені:

1. Налаштування переривань з інтервалом в одну секунду та одну хвилину. Це дозволить консолі перемикатися між зв’язками за допомогою послідовного порту (використовується для зв’язку з блоком ретрансляції або персональним комп’ютером) і модему (використовується для зв’язку з комутаторами моніторів).

2. Передача персонажа з плат на Hyper-terminal [12] на стаціонарному комп'ютері.

3. Наявність символу, що передається та приймається між двома дошками.

4. Передача потоку символів і тим самим пакету від плат до Гіпертерміналу.

5. Передача та отримання пакету між двома платами.

6. Передача пакету читання/запису MODBUS від плат до Гіпертерміналу.

7. Нарешті пакет читання/запису MODBUS передається та приймається між двома платами.

На платах є по три різних світлодіоди, які використовувалися для налагодження та перевірки результатів. Три світлодіоди були класифіковані як «контрольний світлодіод», «модемний світлодіод» і «радіочастотний світлодіод». Для генерації переривань з інтервалом в одну секунду та одну хвилину використовувалися два прапорці, які встановлювалися на основі лічильників таймера. Як і коли прапори були встановлені, світлодіод на платі був налаштований на світіння. Світлодіод буде світитися «зеленим» для кожної секунди, що генерується, і «червоним» для кожної хвилини.

Після створення програмних переривань плата консолі була запрограмована на передачу символу кожну секунду зі швидкістю 19200 Кбіт/с. Для передачі символу була написана функція під назвою «TransmitByte». Після програмування плата була підключена до настільного комп'ютера через comport і з'єднання Hyper-terminal. На малюнку 3.3 показано знімок символу, отриманого вікном Hyper-terminal.

Після цього плата консолі була запрограмована на передачу символу кожну хвилину. На рисунках 3.4, 3.5, 3.6, 3.7 показано символ, отриманий у вікні гіпертерміналу через одну хвилину, 2 хвилини, 5 хвилин і 10 хвилин відповідно. Після успішної передачі символу через необхідні інтервали ретрансляційний блок також було запрограмовано на передачу символів. Після того, як це було зроблено,

блок ретрансляції був запрограмований на отримання персонажа з консолі. Для обробки кожного символу була написана функція під назвою «ReceiveByte». Після цього плата консолі та релейний блок були індивідуально запрограмовані на передачу та прийом символу з кожним інтервалом у одну хвилину відповідно. Потім обидві плати були з'єднані за допомогою кабелю RS-232. У разі успішної передачі символу засвітиться світлодіод.

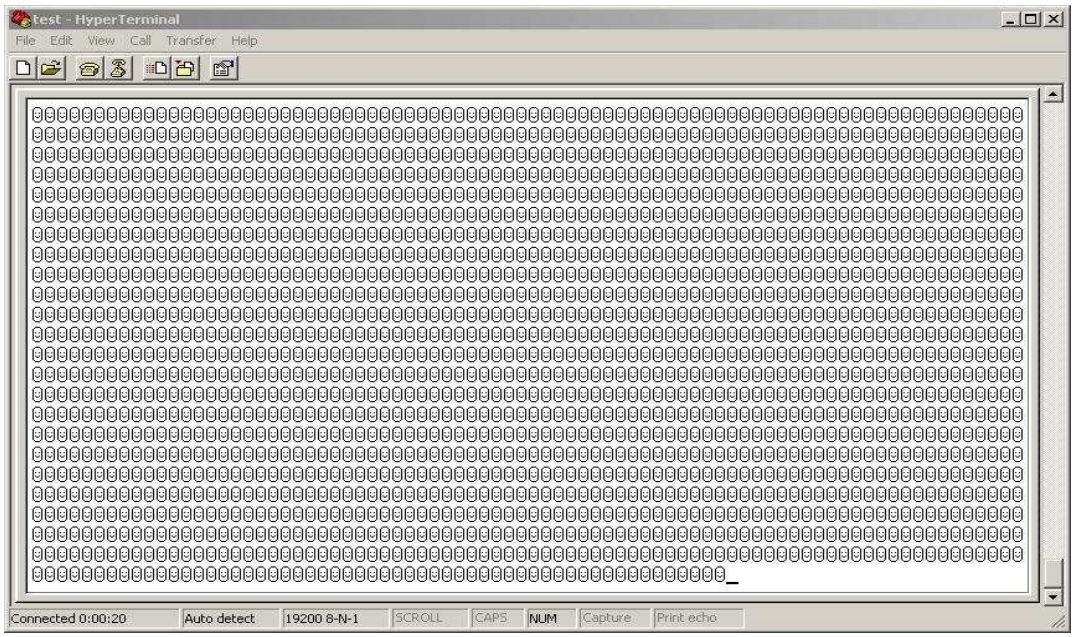


Рисунок 3.3- Вікно Hyper-Terminal отримує символи щосекунди

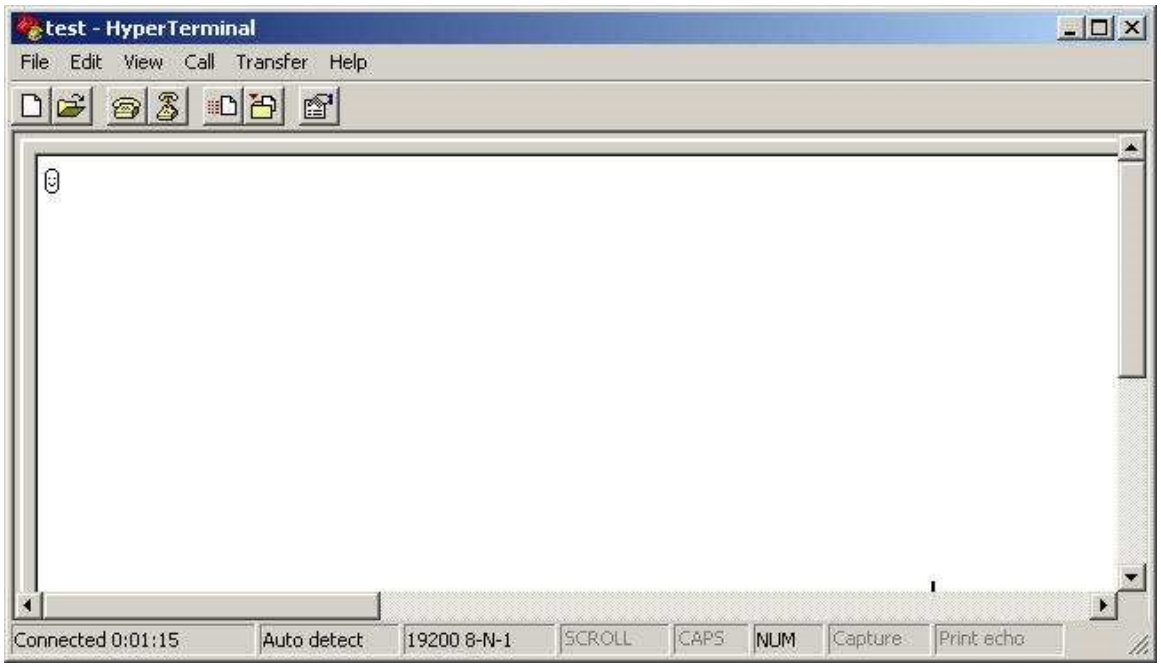


Рисунок 3.4- Вікно гіпертерміналу через одну хвилину

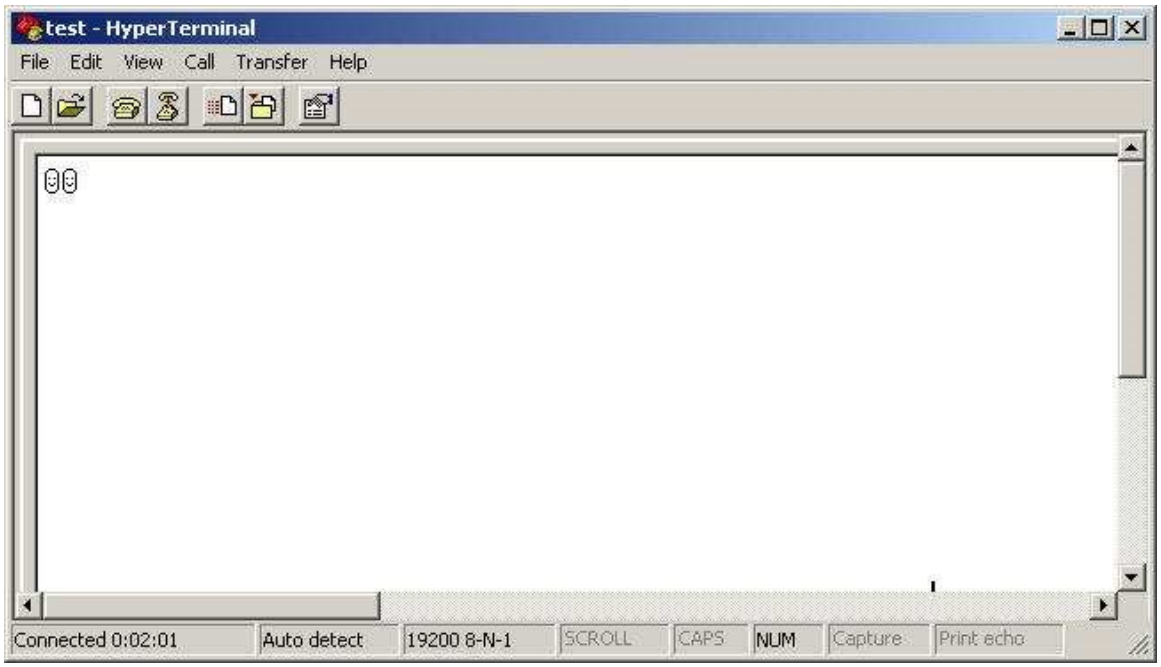


Рисунок 3.5- Вікно гіпертерміналу через 2 хвилини

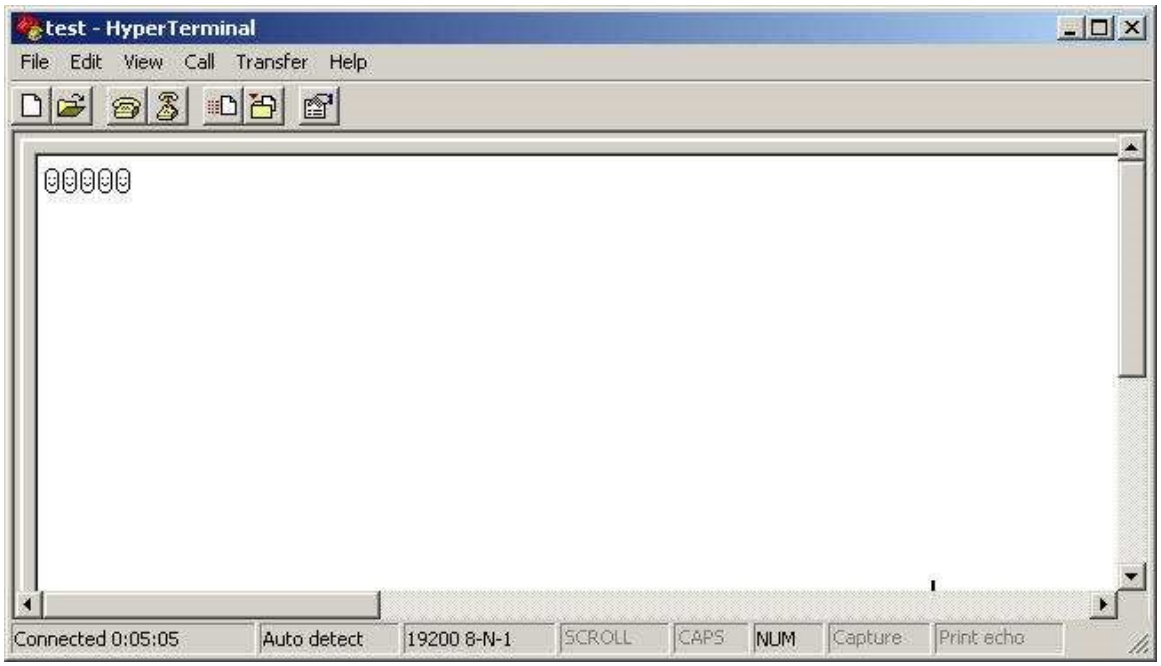


Рисунок 3.6- Вікно гіпертерміналу через 5 хвилин

Консольна плата була налаштована на світилося, а після успішного прийому символу світлодіодний індикатор на релейному блоці був налаштований на світіння. Як тільки це завдання було виконано, наступним кроком було змусити блок ретрансляції надіслати символ відповіді на панель консолі. Логіка блоку реле була знову змінена в програмному забезпеченні для передачі символу відповіді після успішного отримання та обробки символу, надісланого платою консолі. Функція «TransmitByte» була використана в Relay Unit цього разу для надсилання символу відповіді.

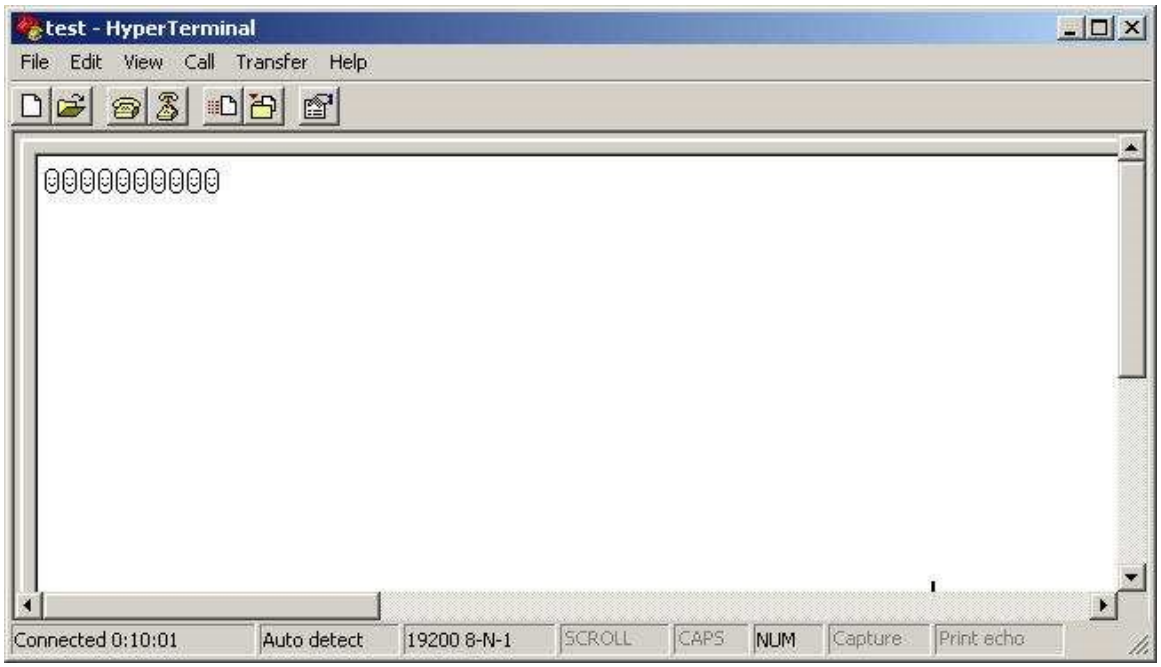


Рисунок 3.7- Вікно гіпертерміналу через 10 хвилин

Ця функція використовувалася після функції «ReceiveByte», тобто лише після перевірки вхідного символу. На платі консолі функція «ReceiveByte» використовувалася для обробки прийому символу відповіді, надісланого блоком ретрансляції. Різні світлодіоди були налаштовані на світіння для кожної дії, що відбувається. Плата консолі передає символ через кожну одну хвилину, що підтверджується світінням світлодіода, а блок ретрансляції приймає символ. Після перевірки світлодіодний індикатор на релейному блоці засвітиться, і після цього релейний блок передасть символ відповіді. Консоль, яка перебувала б у стані «Очікування отримання», виявляла б символ відповіді та світилася світлодіодом, коли отриманий символ було перевірено. Консоль була розроблена для передачі символу з інтервалом у одну хвилину, незалежно від того, отримує чи не отримує символ відповіді. Був встановлений певний таймер, який дозволяв консолі вийти зі стану «Очікування отримання» та повернутися в режим передачі. Консоль використовувала прапорець, який встановлювався, коли будь-який символ було виявлено для отримання. Однак, коли консоль передавала символ, будь-який вхідний символ відхилявся.

Після того, як завдання зв’язку двох пристроїв із символами було виконано, наступним кроком є пакети або потік символів.

довелося здійснити обмін. Спочатку потік з двох символів передавався одночасно як пакет із плати консолі. Це отримувало вікно Гіпертерміналу кожну хвилину. Для передачі потоку символів у вигляді пакету використовувалася функція під назвою «TxPacket». Функція використовувала покажчики та буфери даних для передачі. На рисунках 3.8, 3.9, 3.10 показано вікно гіпертерміналу, яке отримує простий пакет, що складається з двох символів 1 і 2, через одну хвилину, 2 хвилини та 5 хвилин.

Після цього блок ретрансляції було запрограмовано на отримання безперервного потоку символів. Функція під назвою «RxPacket» використовувалася для обробки прийому пакету. Однак, незважаючи на те, що ретрансляційний блок отримав пакет, що складається з 2 символів, дизайну не вистачало логіки визначення кінця пакета. Для цього було встановлено таймер для отримання кожного послідовний байт у пакеті. Цей тайм-аут буде встановлено через 2 мілісекунди, після чого блок ретрансляції не класифікуватиме жодний вхідний символ як частину попереднього пакету. Після того, як це було зроблено, блок ретрансляції було запрограмовано на передачу свого пакету відповіді на консоль.

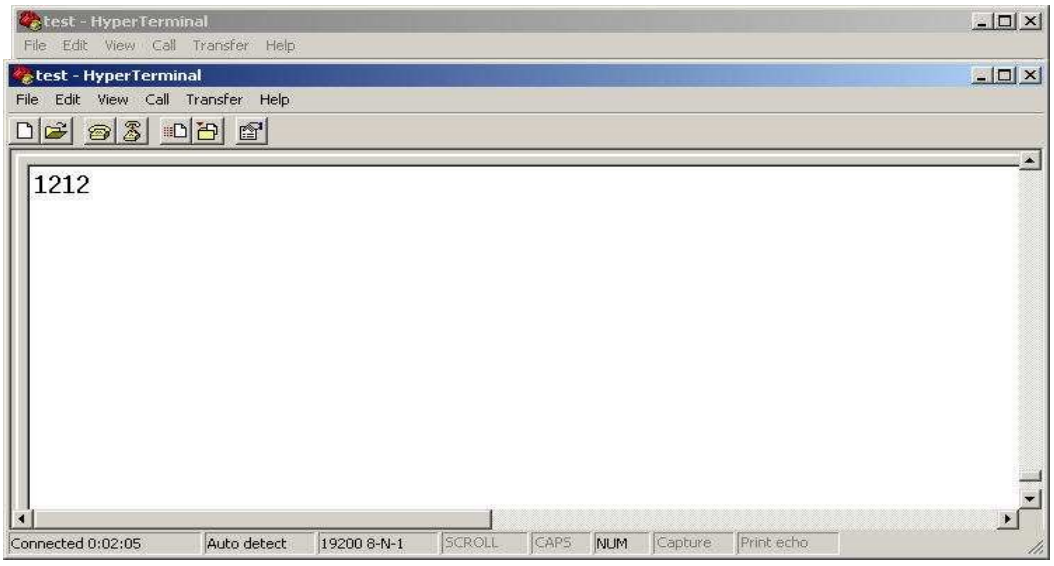


Рисунок 3.8- Вікно гіпертерміналу отримує пакет через 1 хвилину

Рисунок 3.9: Вікно гіпертерміналу отримує пакет через 2 хвилини

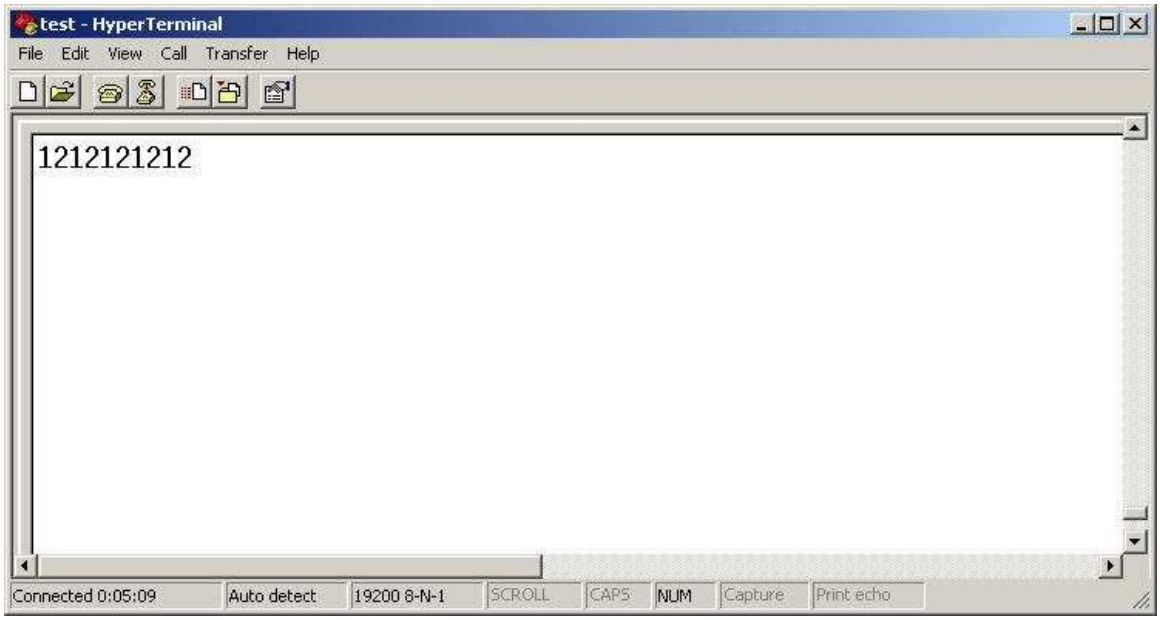


Рисунок 3.10- Вікно гіпертерміналу отримує пакет через 5 хвилин

Функція «TxPacket» тепер використовувалася в модулі ретрансляції для передачі пакету, а функція «RxPacket» разом із тайм-аутами використовувалася в консолі для обробки пакета відповіді від блоку ретрансляції. Блок ретрансляції передав пакет відповіді лише після перевірки пакета, отриманого від Консолі. Консоль також використовувала таймери для прийому та поверталася в режим передачі, якщо будь-який тайм-аут під час прийому стався до отримання всього пакета відповіді. Неповністю отриманий пакет відповіді було відкинуто консоллю, і вона продовжувала передавати пакет через кожну хвилину. Прапор прийому буде встановлено, якщо будуть виявлені будь-які подальші вхідні дані. Однак, якщо консоль передає в цей момент часу, вхідні дані відхиляються. Ця логіка також була реалізована в Relay Unit.

Будь-які вхідні дані було відхилено, коли вони передавали пакет. Як і в попередньому випадку передачі та обміну одним символом, світлодіоди були налаштовані на світіння для кожної з виконуваних дій. Початкова передача пакета з консолі, прийом ретранслятором, перевірка отриманого пакета та подальша передача пакета відповіді та, нарешті, прийом і перевірка пакета відповіді

Консоль позначалася світінням різних світлодіодів на кожній з плат.

Після того, як обмін даними в пакетній формі було завершено, наступним кроком був вирішальний сегмент роботи. Протокол послідовної лінії MODBUS мав

бути реалізований на пакетах даних, і це передбачало розробку кадрів для виконання різних функцій.

Вперше був розроблений пакет читання MODBUS, який показаний таблиці 3.1. Пакет читання MODBUS починався з адреси підпорядкованого пристрою, яка складалася з одного байта «0x01», який у цьому випадку відноситься до блоку ретрансляції. За ним йшов код функції, який знову був байтом. Використаний код функції був одним із «0x03», «0x04» і «0x07» для функції читання. Функція

За кодом слідували старший байт початкової адреси та молодший байт початкової адреси.

Таблиця 3.1

Пакет читання MODBUS

Slave Address	Function Code	Start Address Hi	Start Address LO	No of Points Hi	No of Points Lo	CRC Lo	CRC Hi

Ці два байти визначали адресу, з якої блок реле або персональний комп'ютер мав зчитувати необхідні дані. За молодшим байтом початкової адреси слідували старший байт кількості точок і молодший байт кількості точок. Ці два байти забезпечували кількість бітів, які мали бути прочитані блоком реле або персональним комп'ютером. За молодшим байтом кількості точок слідували молодший байт CRC і старший байт CRC. Функція «create_req_packet» була розроблена для кадрування пакета читання, а функція «TransmitByte» використовувалася для передачі зчитуваних кадрів до Hyper-Terminal. Консоль була запрограмована на передачу пакета читання MODBUS щохвилини та підключена до настільного комп'ютера. На рисунках 3.12 і 3.11 показано, як Hyper-Terminal отримує зчитаний пакет від консолі через одну хвилину і 2 хвилини. Hyperterminal перетворює шістнадцятковий вміст прочитаного пакету в символи ASCII і відображає їх. Таким чином, символи «130:PL» у вікні представляють шістнадцяткові значення «31,33,30,3A,30,50,4C,60» відповідно. Під час передачі пакету шістнадцяткові значення перетворюються на ASCII шляхом додавання значення «0x30» до кожного байта. Таким чином, значення, отримані у вікні Hyperterminal, представляють шістнадцяткову послідовність «0x01, 0x03, 0x00, 0x0A, 0x00, 0x20, 0x1C, 0x12».

Перший байт «0x01» вказує на адресу підлеглого, за яким слід читання коду функції «0x03». У цьому випадку консоль намагається прочитати параметр у 16-му байті адреси, який позначається початковою адресою «0x00 0x0A». Кількість точок або бітів для зчитування становить 32, які визначаються байтами кількості точок «0x00 0x20». Останні 2 байти «0x1C» і «0x12» є байтами CRC, які обчислюються перед передачею кадру зчитування. Функція «calc_crc», яка використовується для обчислення CRC, викликається функцією «create_req_packet» до початку передачі. CRC виконується для кожного вмісту повідомлення кадру, а кінцевий результат додається в кінці кадру.

У гіпертерміналі наступним очевидним кроком було впоратися з прийомом зчитування пакет на блоці ретрансляції. Щоб впоратися з цим завданням, розмір приймального буфера було збільшено для розміщення пакета, розмір якого тепер був більшим. Блок ретрансляції перейде в режим прийому, як тільки буде виявлено початковий символ. Перший символ, тобто отримана адреса веденого пристрою, перевіряється блоком ретрансляції, щоб визначити, чи призначений пакет для нього. Після перевірки підпорядкованої адреси блок ретрансляції починає приймати решту пакету, починаючи з функціонального коду. Оскільки отриманий пакет є пакетом для читання, блок ретрансляції було налаштовано з лічильником байтів або покажчиком, який дозволяв йому отримувати лише 6 байт, що залишилися, тобто старші та молодші байти початкової адреси, кількість точок старшого та молодшого байтів і Старший і молодший байти CRC.

Будь-які додаткові байти, отримані після цього моменту часу, були відкинуті. Потім блок ретрансляції перевіряє весь кадр, обчислюючи CRC для кожного вмісту кадру, за винятком останніх двох байтів. Обчислений результуючий CRC перевіряється за допомогою двох байтів CRC, надісланих у вихідному кадрі повідомлення, і перевіряється. Як і раніше, про передачу, прийом і перевірку пакета свідчить світіння світлодіодів на обох платах. Блок ретрансляції використовується з таймерами, які використовують обмеження часу на послідовне отримання байтів, а також на загальний час, протягом якого повинен бути отриманий весь кадр.

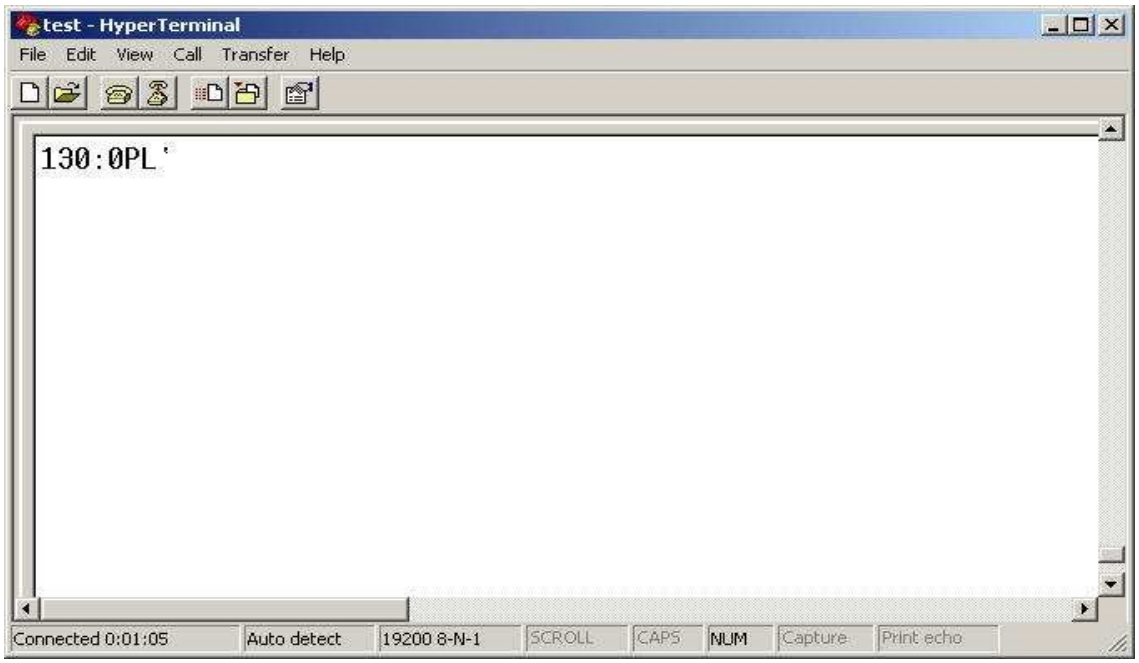


Рисунок 3.11: Гіпертермінал отримує пакет читання MODBUS через 1 хвилину

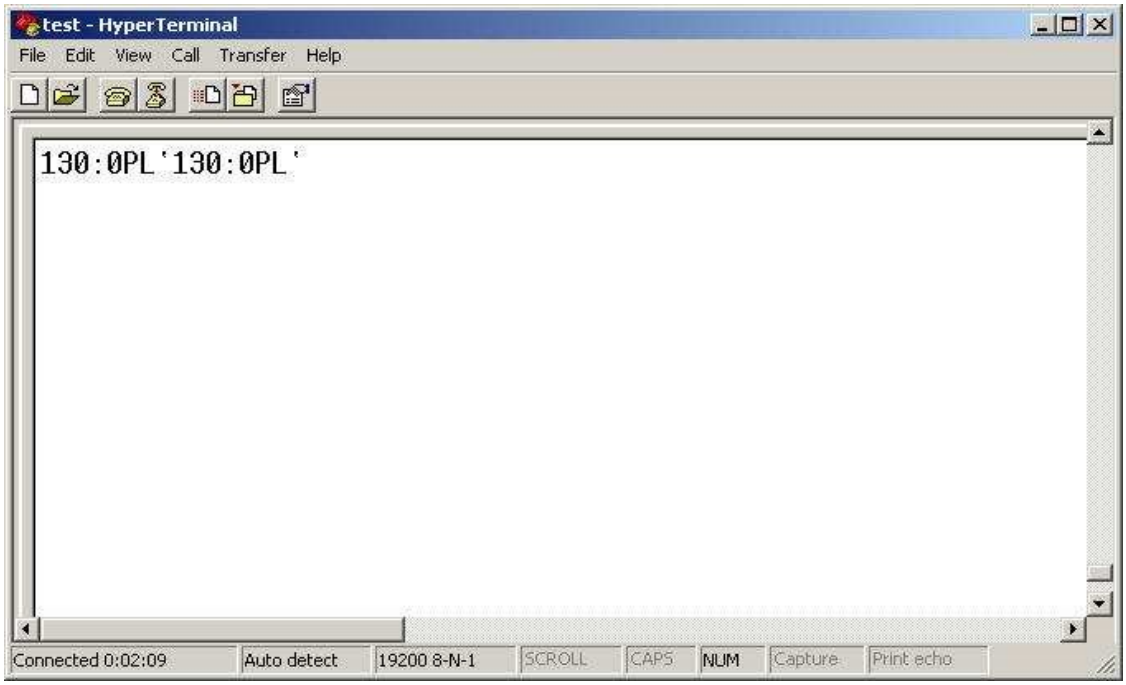


Рисунок 3.12: Гіпертермінал отримує пакет читання MODBUS через 2 хвилини

Якщо під час прийому послідовних байтів виникає будь-який тайм-аут, прийом припиняється, а кадр відхиляється, якщо він не був отриманий повністю. Кадр також скидається, коли таймер, який контролює прийом усього кадру, закінчується, якщо кадр не отримано повністю до цього часу.

Після успішної передачі пакета зчитування від консолі до ретранслятора потрібно було зробити наступний крок, щоб ретранслятор надіслав пакет відповіді

із запитаними даними. Карта пам'яті структури танка була реалізована в пульті і блоці реле. Щоб надіслати пакет відповіді, була розроблена функція «create_resp_pack», яка створила відповідну відповідь на запитувану дію. Пакет відповіді буде створений після виконання запитуваної дії ретранслятором. В таблиці 3.2 показано структуру відповідного пакета в блоці ретрансляції. Пакет відповіді містить адресу підлеглого пристрою «0x01», тобто адресу блоку ретрансляції, за яким слідує код функції «0x03», який вказує на те, що спочатку було запитано дію читання. За кодом функції слідує байт підрахунку байтів, який містить кількість байтів, які зчитуються, починаючи з адреси, зазначеної байтами початкової адреси в пакеті запиту. За підрахунком байтів слідують старші та молодші байти даних, які є даними, які запитує консоль. За байтами даних йдуть старший і низький байти CRC, які обчислюються реле. CRC виконується для всього вмісту частини кадру, а отримані два байти обчислення додаються в кінці кадру повідомлення.

Після того, як Relay було запрограмовано на надсилання пакету відповіді до Консоль, дві плати були підключені та перевірені на належну роботу. Консоль надсилає пакет читання MODBUS щохвилини, запитуючи прочитати 16-й байт із ретранслятора. Ретранслятор отримав пакет і виконав необхідну дію та надіслав консолі необхідні дані.

Таблиця 3.2

Пакет відповіді на читання MODBUS

Slave Address	Function Code	Byte Count	Data Hi	Data Lo	CRC Lo	CRC Hi

Після цього консоль перевірила отриманий пакет відповіді. Перевірка включала зчитування даних, а також перевірку CRC. Обчислення CRC і перевірка, які виконує консоль, подібні до обчислень, які виконує блок ретрансляції, коли він отримує пакет. Усі ці дії були підтверджені та перевірені світлодіодами на кожному кроці.

Після двостороннього обміну читанням було виконано останній крок передавання пакету запису MODBUS від консолі до ретранслятора та відповідної відповіді, надісланої від ретранслятора до консолі. В таблиці 3.3 показано дизайн

пакета запису MODBUS. Пакет запису MODBUS почався з адреси веденого пристрою «0x01», що відноситься до блоку ретрансляції. Після цього йшов байт коду функції, який позначав функцію запису. Код функції може бути одним із «0x05», «0x06», 0x0F і «0x10». Після функціонального коду йшли старший і молодший байти початкової адреси, які вказували блоку ретрансляції адресу, з якої повинні були записуватися дані.

Таблиця 3.3

Пакет запису MODBUS

Slave Address	Function Code	Start Address Hi	Start Address LO	No of Points Hi	No of Points Lo	Byte Count	Data Hi	Data Lo	CRC Lo	CRC Hi

Цей підрахунок байтів, хоча може здатися непотрібним, відіграє життєво важливу роль у частині процесу прийому. Поле підрахунку байтів вказує блоку ретрансляції кількість байтів даних, яку він має очікувати для подальшого прийому. Кількість байтів — це сума кількості байтів Data High і Data Low, які йдуть після поля Byte Count у кадрі повідомлення. Байти CRC, які обчислюються для кожного вмісту повідомлення, додаються в кінці повідомлення.

Як і в попередньому випадку передачі пакета для читання, функція «create_req_pack» використовувалася для кадрування пакета запису, а «TransmitByte» використовувався для передачі пакета запису від консолі до гіпертерміналу на настільному комп’ютері. На рисунках 3.13 і 3.14 показано пакет запису, отриманий у вікні гіпертерміналу через одну та дві хвилини відповідно. Символи ASCII «150:0P412342=» у вікні представляють шістнадцяткові символи «31, 35, 30, 3A, 30, 50, 34, 31, 32, 33, 34, 32, 3D» відповідно. Під час передачі пакету шістнадцяткові значення перетворюються на ASCII шляхом додавання значення «0x30» до кожного байта. Таким чином, значення, отримані у вікні Hyperterminal, представляють шістнадцяткову послідовність «0x01, 0x05, 0x00, 0x0A, 0x00, 0x20, 0x04, 0x01, 0x02, 0x03, 0x04, 0x02, 0x13».

Рисунок 3.17: Вікно гіпертерміналу отримує пакет запису MODBUS через 2 хвилини

Код функції «0x05». У цьому випадку консоль просить записати значення параметра в 16-й байт адреси, який позначається початковою адресою «0x00».

Перший байт «0x01» вказує адресу підлеглого, а потім читання

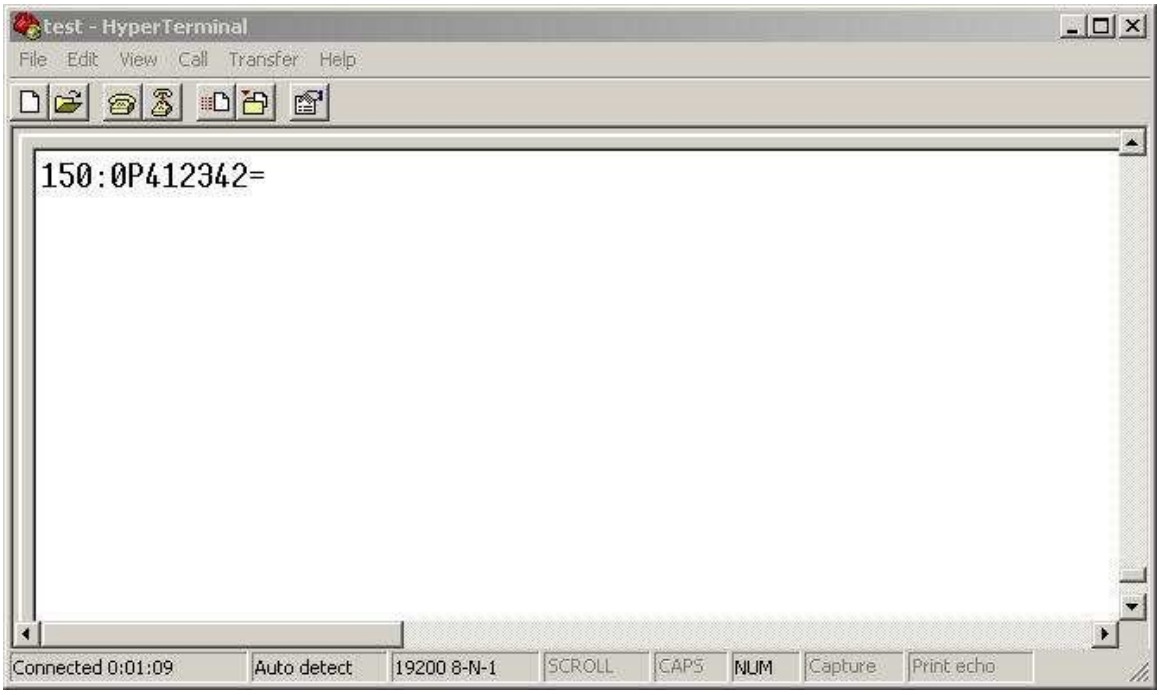


Рисунок 3.13- Вікно гіпертерміналу отримує пакет запису MODBUS через 1 хвилину

0x0A'. Кількість точок або бітів, які потрібно записати, становить 32, які визначаються кількість балів у байтах «0x00 0x20». Загальна кількість байтів у поля Data High і Low мають значення «0x04», що вказується полем «Число байтів». За полем підрахунку байтів ідуть байти даних, які потрібно записати. Байти даних у цьому випадку: «0x01 0x02 0x03 0x04». За цими байтами даних йдуть низький і старший байти CRC, у цьому випадку «0x02» і «0x13». Як і раніше, CRC обчислюється за допомогою функції «calc_crc», яка обчислює CRC для всього вмісту повідомлення. Отримані два байти додаються в кінці кадру.

Після успішної передачі пакета запису від консолі до гіпертерміналу було виконано завдання, що залишилося, щоб блок ретрансляції отримав пакет запису та виконав необхідні операції запису. Блок ретрансляції перейде в режим прийому, як тільки буде виявлено початковий символ. Перший символ, тобто отримана

адреса веденого пристрою, перевіряється блоком ретрансляції, щоб визначити, чи призначений пакет для нього. Після перевірки підлеглої адреси,

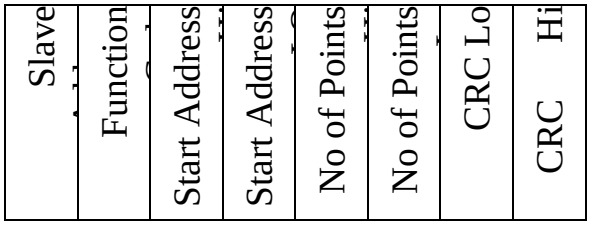
Блок ретрансляції починає приймати решту пакету, починаючи з коду функції. Після визначення пакета як пакета запису блок ретрансляції прийме решту пакета, починаючи зі старшого байта початкової адреси. Він прийматиме байти, доки не з’явиться поле підрахунку байтів. Лічильник використовувався для відстеження прийому вхідних байтів і, як тільки лічильник показував

отримання поля Byte Count, інший лічильник даних або покажчик отримує значення, отримане з поля Byte Count. Після цього Relay зберігатиме вхідні байти в буфері даних, доки лічильник даних не стане нульовим. Коли лічильник даних стає нульовим, що вказує на кінець байтів даних, ретрансляційний блок отримує лише наступні два байти, тобто старший і низький байти CRC, які пізніше використовуються для порівняння та перевірки. Блок ретрансляції обчислює CRC для всього кадру повідомлення, за винятком останніх двох байтів, і результуючий CRC порівнюється з вихідним CRC, доданим до кадру вхідного повідомлення. Кадр буде відкинуто, якщо два набори значень не збігаються один з одним. Інакше буде виконано необхідну дію запису. Як і раніше, про передачу, прийом і перевірку пакета свідчить світіння світлодіодів на обох платах.

Після того, як це було зроблено, блок ретрансляції був розроблений для надсилання пакета відповіді на пакет запису, який він отримав від консолі. Для цього була розроблена функція під назвою «create_resp_pack», яка створює пакет відповіді. В таблиці 3.4 показано структуру пакета відповіді на запис від блоку ретрансляції. Пакет відповіді на запис починається з адреси підлеглого пристрою, тобто «0x01», за яким слідує код функції, який є «0x05» для функції запису. За кодом функції йдуть старший і молодший байти початкової адреси, які позначають початкову адресу в пам’яті, з якої були записані дані. За байтами початкової адреси йдуть старші та молодші байти кількості точок, які надають інформацію про кількість записаних бітів в пам'ять релейного блоку. Як і в попередніх випадках, за молодшим байтом кількості точок йдуть молодший і старший байти CRC, які обчислюються ретрансляцією для всього вмісту кадру та додаються в кінці.

Таблиця 3.4

Пакет відповіді MODBUS на запис



Після того, як Relay було запрограмовано для надсилання пакету відповіді, 2 плати були з’єднані та спостерігався обмін пакетами запису. Покрокові дії, які виконуються, починаючи з передачі консоллю пакета запису, ретранслятора, який приймає його та перевіряє отримані дані, ретранслятор відповідає на консоль і, нарешті, консоль перевіряє пакет відповіді, що вказується світінням світлодіодів.

Консоль після отримання пакета відповіді, незалежно від того, чи це відповідь на запит на читання або запис, надісланий раніше, перевірить адресу підпорядкованого пристрою, щоб визначити, чи надійшла відповідь від правильного пристрою. Після цього він почне приймати решту пакета, починаючи з коду функції. Консоль була налаштована з лічильниками даних або покажчиками, які перевіряли решту вхідних даних. Використовувані покажчики даних були різними для випадків читання та запису. Це було тому, що розміри пакетів відповіді для читання та запису відрізнялися. Як тільки покажчики даних стали нульовими, він не прийматиме жодних додаткових даних. Операція отримання консолі була подібна до операції ретранслятора. Коли він повертався до передачі нового пакету, будь-які вхідні дані відхилялися. Консоль також мала таймери, які перевіряли отримання послідовних байтів, а також отримання всього кадру повідомлення.

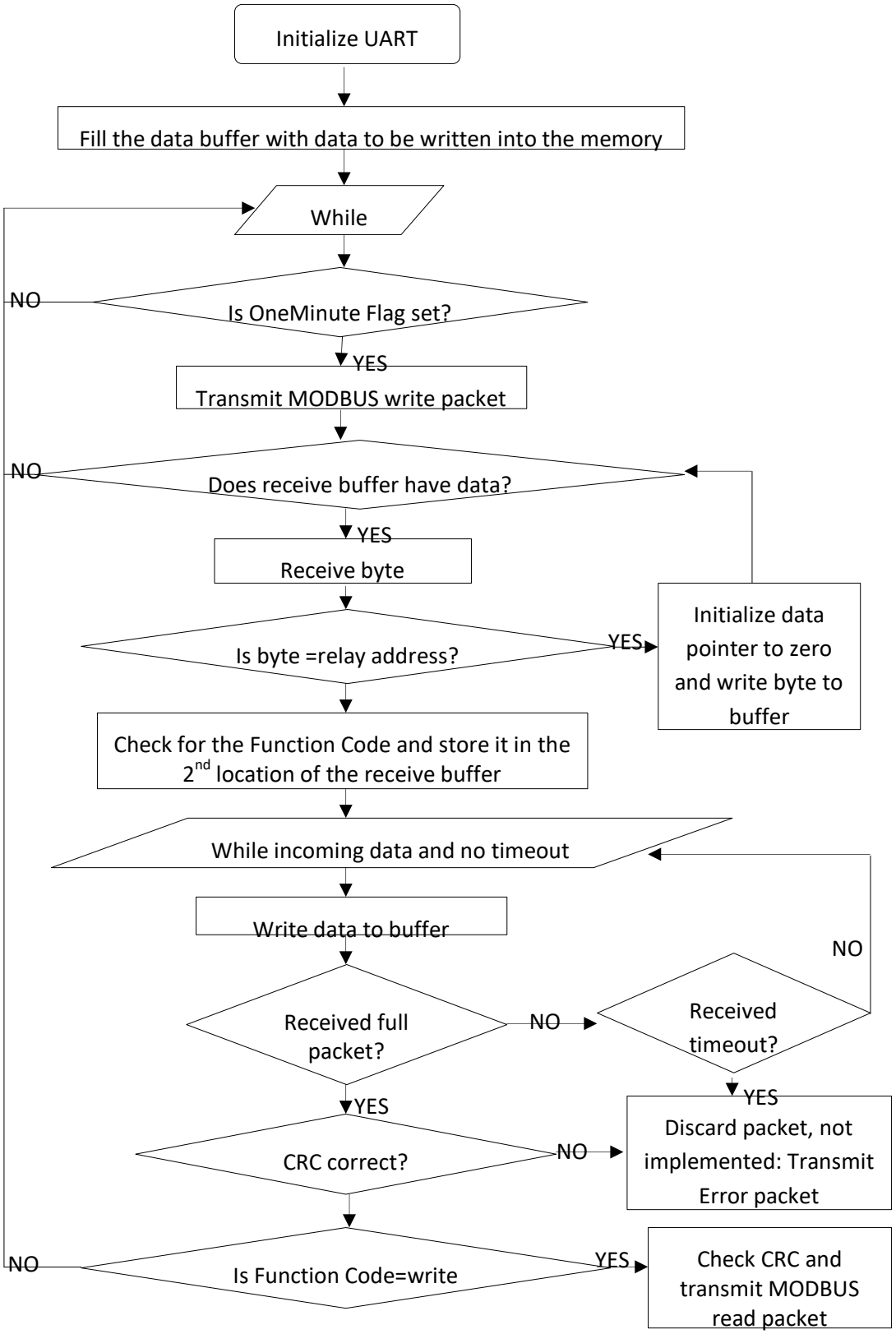


Рисунок 3.14- Потік кінцевих дій консолі

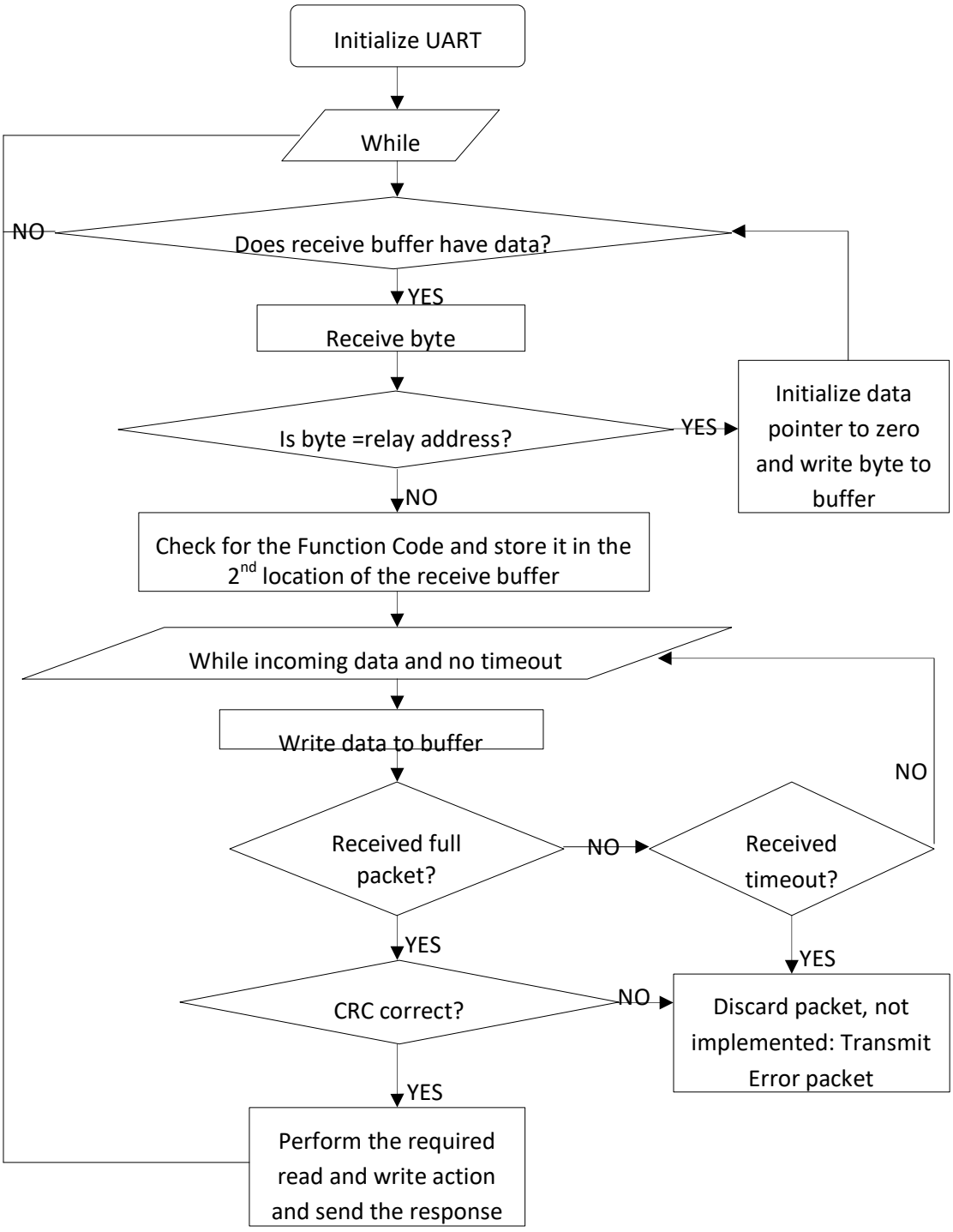


Рисунок 3.15- Потік кінцевих дій блоку реле

Якщо будь-який із цих таймерів закінчився до того, як було отримано все повідомлення, пакет буде відкинуто.

Після успішної обробки обміну пакетами запису між двома платами було реалізовано додаткове завдання. Консоль ініціює запит на запис до блоку ретрансляції, який обробить запит і відповість пакетом відповіді. Після перевірки відповіді консоль ініціює запит на читання, який намагається прочитати ті самі дані з місця пам’яті, яке було записане попереднім запитом на запис. Блок ретрансляції

обробить запит на читання та відповість із відповідними даними. Після отримання відповіді Консоль перевірить дані. Цей процес гарантував, що запит на запис, ініційований на початку, точно виконується для відповідного параметра та у відповідній ділянці пам’яті. На рисунках 3.14 і 3.15 показано кінцевий потік дій консолі та релейного блоку.

Основне завдання зв’язку консолі з блоком ретрансляції та персональним комп’ютером було виконано. Зв'язок базувався на архітектурі клієнт-сервер і був реалізований за допомогою послідовного каналу RS 232 між окремими об'єктами. На моделі успішно реалізовано протокол MODBUS.

Було перероблено всю модель і реалізовано відповідне програмне забезпечення на агрегатах. Логіка програмного забезпечення була перероблена, написана та перевірена шляхом успішного програмування плат. Було модифіковано існуюче раніше програмне забезпечення та написані функції для створення, передачі та прийому пакету MODBUS. Оскільки дані були записані в EEPROM блоків, у разі будь-якого збою живлення система поверталася до останньої конфігурації та налаштувань реле. Однак у цієї роботи є обмеження щодо продуктивності.

MODBUS, як обговорювалося в попередніх розділах, дозволяє передати або отримати загалом 256 байт в одному пакеті в одній транзакції. Цілком очевидно, що пакет запису MODBUS буде найбільшим розміром пакета в одній транзакції між пристроями. Пакет запису включатиме адресу підпорядкованого пристрою, код функції, початкову адресу, кількість точок, кількість байтів, які займають 7 байтів. Таким чином, кількість даних, які можуть бути записані шляхом передачі одного пакета запису, буде обмежена 245 байтами. Подібним чином дані, які можуть бути передані в зчитуваному пакеті MODBUS, обмежені 251 байтом через кадрування пакета. Якщо кількість резервуарів і відповідна кількість параметрів, які потрібно обробляти, збільшуються, тоді буде потрібно більше однієї транзакції MODBUS за раз. Хоча це не вплине на ефективність або пропускну здатність системи, це потребуватиме більше обчислень і транзакцій. Крім того, можуть виникнути проблеми з шумом, якщо два пристрої розташовані далеко один від одного та з’єднані за допомогою кабелю RS-232.

Майбутній обсяг цієї роботи включатиме впровадження кодів винятків. Усі коди винятків, визначені MODBUS, повинні бути реалізовані в моделі.

Розгорнута схема взаємодії між пристроями за допомогою протоколу Modbus:

1. Ініціалізація:

- Майстер (Master) ідентифікує пристрої, які беруть участь у взаємодії.
- Встановлюється фізичне з'єднання між майстром і ведучими (Slave) пристроями.

2. Запит від майстра:

- Майстер формує запит на читання або запис даних та вказує адресу ведучого пристрою та регістр, з якого чи на який будуть зчитані/записані дані.
- Створюється Modbus-пакет, який містить інформацію про функцію, адресу, кількість даних та контрольну суму.

3. Відповідь ведучого пристрою:

- Ведучий пристрій (Slave) отримує запит та аналізує його.
- Ведучий формує відповідь, яка містить зчитані дані або підтвердження про успішне виконання запису.
- Modbus-пакет відправляється майстру.

4. Обробка відповіді майстра:

- Майстер отримує відповідь і аналізує її на предмет помилок чи відповідності запиту.
- Виконує необхідні дії на основі отриманих даних або підтвердження.

5. Завершення обміну:

- Закривається фізичне з'єднання між майстром і ведучими пристроями.
- Майстер може ініціювати новий обмін даними або припинити взаємодію.

Ця схема дозволяє вам візуалізувати послідовність подій під час обміну даними між пристроями за допомогою протоколу Modbus. Ви можете використати цей опис для створення графічної схеми за допомогою спеціалізованих інструментів для малювання.

Визначення ролей: майстер та ведучий, об'єкти даних в контексті протоколу Modbus:

1. Майстер (Master):

- Опис ролі:
  - Майстер є ініціатором обміну даними в системі, він керує процесом та ініціює запити до ведучих пристроїв.
  - Майстер може взаємодіяти з одним або кількома ведучими пристроями одночасно.
- Функції майстра:
  - Створення та відправка запитів на зчитування або запис даних до ведучих пристроїв.
  - Обробка відповідей від ведучих пристроїв.
  - Керування та координація обміну даними в мережі.

2. Ведучий (Slave):

- Опис ролі:
  - Ведучий пристрій є об'єктом, який відповідає на запити майстра та забезпечує обмін даними між фізичними або логічними пристроями.
  - Ведучий може бути реалізований у вигляді датчиків, контролерів, плат управління тощо.
- Функції ведучого:
  - Прийом та розкодування запитів від майстра.
  - Формування відповідей на запити майстра.
  - Забезпечення доступу до об'єктів даних та їх зміна відповідно до отриманих запитів.

3. Об'єкти даних:

- Опис:
  - Об'єкти даних представляють собою конкретні значення або параметри, які майстер може зчитувати або змінювати в ведучих пристроях.
  - Може включати інформацію про стан обладнання, вимірювані значення, конфігураційні дані та інше.
- Типи об'єктів даних:

- Входні (Input) регістри: Призначені для зчитування майстром.
- Виходні (Output) регістри: Призначені для запису майстром.
- Регістри холдингу (Holding): Дозволяють як зчитування, так і запис.
- Регістри дискретних входів (Discrete Inputs): Вказують на стан конкретних подій чи обставин.

Це визначення ролей та об'єктів даних допомагає уявити структуру взаємодії між майстром і ведучими пристроями в системі, яка використовує протокол Modbus.

Обмін даними за допомогою протоколу Modbus включає кілька етапів, що відбуваються між майстром та ведучими пристроями. Нижче наведено пояснення кожного етапу:

1. Ініціалізація та Підготовка:

- Майстер: Майстер ініціює обмін даними, ідентифікує ведучі пристрої в мережі та встановлює фізичне з'єднання.
- Ведучий: Ведучі пристрої готуються до обробки запитів та чекають на команди від майстра.

2. Формування та Відправка Запиту:

- Майстер: Створює Modbus-запит, включаючи функцію, адресу ведучого пристрою, регістр та інші параметри.
- Ведучий: Відслідковує мережевий трафік та очікує на Modbus-запит від майстра.

3. Обробка Запиту ведучого:

- Ведучий: Приймає та розкодовує Modbus-запит, визначає, які дії необхідно виконати (зчитати або записати дані) та підготовлюється до відповіді.

4. Відповідь ведучого:

- Ведучий: Створює Modbus-відповідь, включаючи зчитані дані або підтвердження про успішний запис.
- Майстер: Приймає відповідь від ведучого та аналізує її для подальшої обробки.

5. Обробка та Підготовка до Нового Запиту:

- Майстер: Обробляє відповідь ведучого, визначає наступні кроки та

готується до нового запиту або завершення обміну даними.

- Ведучий: Готується до нового Modbus-запиту або очікує завершення обміну.

6. Завершення та Розірвання З'єднання:

- Майстер: При необхідності завершує обмін даними та розривається фізичне з'єднання з ведучими пристроями.
- Ведучий: Завершує обробку запиту та готується до нових можливих запитів.

Ці етапи представляють собою цикл обміну даними між майстром та ведучими пристроями, який може повторюватися в залежності від потреб системи. Кожен етап включає в себе взаємодію між майстром та ведучим пристроєм для ефективного обміну інформацією через протокол Modbus.

Детальний алгоритм обміну даними за допомогою протоколу Modbus включає кілька етапів та процедур. Нижче подано алгоритм, який описує обмін даними між майстром (Master) і ведучим (Slave) пристроєм:

1. Ініціалізація:

- Майстер ідентифікує ведучі пристрої в мережі.
- Ведучі пристрої готуються до обробки запитів.

2. Формування Modbus-запиту:

- Майстер створює Modbus-запит, включаючи:
  - Адресу ведучого пристрою.
  - Код функції (читання/запис даних).
  - Адресу початкового регістру або біта.
  - Кількість регістрів або бітів для читання чи запису.
  - Контрольну суму для забезпечення цілісності.

3. Відправка Modbus-запиту:

- Майстер відправляє Modbus-запит ведучому пристрою через фізичне з'єднання (зазвичай RS-485, TCP/IP тощо).

4. Прийом та Розкодування Modbus-запиту ведучим:

- Ведучий пристрій приймає Modbus-запит.
- Розкодовує отриманий запит і визначає, які дані потрібно читати або

записувати.

5. Обробка та Відповідь ведучого:

- Ведучий пристрій виконує дії відповідно до отриманого запиту.
- Формує Modbus-відповідь, включаючи необхідні дані або підтвердження про успішний запис.

6. Відправка Modbus-відповіді:

- Ведучий відправляє Modbus-відповідь майстру через фізичне з'єднання.

7. Прийом та Розкодування Modbus-відповіді майстром:

- Майстер приймає Modbus-відповідь від ведучого.
- Розкодовує отриману відповідь і визначає, чи був запит успішно виконаний.

8. Обробка та Аналіз Результатів:

- Майстер обробляє отримані дані або аналізує підтвердження успішності запису.
- При необхідності майстер може ініціювати новий запит чи завершити обмін даними.

9. Завершення Обміну:

- Майстер і ведучий готуються до нового обміну даними або завершують з'єднання, якщо обмін завершено.

Цей алгоритм визначає послідовність операцій між майстром та ведучим пристроєм під час обміну даними за допомогою протоколу Modbus. Кожен крок має свої власні характеристики та процедури, що визначають коректну взаємодію між пристроями.

Розгляд асинхронного та синхронного обміну даними є важливою частиною розуміння протоколу Modbus і взагалі концепцій обміну даними в комп'ютерних мережах.

1. Синхронний обмін:

- Характеристика:
  - Обмін даними відбувається за строго визначеними часовими рамками.
  - Кожна сторона очікує завершення операції перед тим, як розпочати

наступну.

- Переваги:
  - Простіший в реалізації та розумінні.
  - Легше виявляти та виправляти помилки.
- Недоліки:
  - Може викликати затримки у випадку надто довгих операцій.
  - Не ефективний в умовах мереж із змінною затримкою.

2. Асинхронний обмін:

- Характеристика:
  - Обмін даними не пов'язаний із строго визначеними часовими рамками.
  - Сторони можуть виконувати операції незалежно одна від одної.
- Переваги:
  - Дозволяє оптимально використовувати ресурси.
  - Підходить для великих систем та систем із змінною затримкою.
- Недоліки:
  - Складніше відслідковувати та виправляти помилки через асинхронність.
  - Вимагає виваженої синхронізації для уникнення конфліктів.

Застосування у протоколі Modbus:

- Асинхронний обмін:
  - Зазвичай використовується в мережах із значними затримками, де синхронізація важко досяжна.
  - Ведучий пристрій може відповідати на запити незалежно від часових рамок майстра.
- Синхронний обмін:
  - Зручний у використанні для простих систем або в мережах із низькою затримкою.
  - Кожен обмін починається тільки після завершення попереднього, що полегшує синхронізацію.

Вибір між асинхронним та синхронним обміном залежить від конкретних вимог системи та умов мережі. У багатьох випадках, особливо в складних мережах,

може використовуватися комбінований підхід, де частину обміну виконують асинхронно, а частину синхронно.

Врахування та обробка помилок в обміні даними за допомогою протоколу Modbus є критично важливим аспектом для забезпечення надійності та стійкості системи. Нижче подано загальні кроки та стратегії для врахування та обробки можливих помилок:

1. Контрольні суми та Перевірка Цілісності Даних:

- Використовуйте контрольні суми або хеш-функції для перевірки цілісності передачі даних. Це дозволяє виявляти помилки, що виникають під час передачі.

2. Обробка Помилок на Рівні Протоколу:

- Використовуйте коди помилок, які передбачені протоколом Modbus, для ідентифікації та визначення типу помилки.
- Обробляйте відповідні винятки або статуси для виправлення або інформування користувача/системи про помилку.

3. Захист від Дублювання:

- Враховуйте можливість дублювання пакетів та використовуйте механізми для виявлення та виправлення дублювання команд.

4. Таймаут та Виявлення Втрати З'єднання:

- Встановлюйте таймаути для очікування відповіді ведучого.
- Перевіряйте на втрату з'єднання та ініціюйте перепідключення або інші заходи в разі втрати комунікації.

5. Журналювання та Логування:

- Реєструйте помилки та інші події для подальшого аналізу та діагностики.
- Виводьте інформацію про помилки до лог-файлів для подальшого вивчення.

6. Механізми Повторюваності та Відновлення:

- Реалізуйте механізми для повторення або відновлення обміну даними в разі помилок.

- Використовуйте буферизацію та кешування для збереження даних та відновлення стану після помилки.

7. Системи Захисту та Автентифікації:

- Застосовуйте методи захисту та автентифікації для попередження атак та забезпечення безпеки обміну даними.

Врахування цих аспектів допомагає створити надійну та стійку систему обміну даними через протокол Modbus. Залежно від конкретних вимог та контексту використання можуть бути визначені специфічні стратегії обробки помилок.

Як мову програмування та середовище розробки я би вибрав Python та Visual Studio Code (VSCode). Нижче подано короткий опис вибору та переваг цих інструментів:

- Python: високорівнева, інтерпретована мова програмування, яка є дуже зручною для розробки та володіє читабельним синтаксисом. Має потужну спільноту та велику кількість бібліотек, зокрема для роботи з мережами та реалізації протоколу Modbus.
- Visual Studio Code (VSCode): легкий, швидкий та розширюваний редактор коду, розроблений Microsoft. Має потужні функції, такі як вбудований відлагоджувальник, підтримка розширень, вбудована система керування версіями, інтегрований термінал і багато інших.

Вибір Python та VSCode:

1. Підтримка Modbus:

- Python має багато бібліотек, таких як PyModbus, які роблять взаємодію з протоколом Modbus легкою та ефективною.
- VSCode має розширення для підтримки Python, що полегшує написання, відлагодження та тестування коду.

2. Швидкість Розробки:

- Python дозволяє швидко розробляти та тестувати рішення, що особливо важливо для ефективного взаємодії з протоколом Modbus.
- VSCode надає інструменти для швидкого написання коду, відлагодження та перевірки синтаксису.

3. Спільнота та Екосистема:

- Python має велику та активну спільноту розробників, що дозволяє легко знаходити рішення на форумах та репозиторіях.
- VSCode також має широку спільноту та велику кількість розширень для різних мов програмування, включаючи Python.

4. Кросплатформенність:

- Як Python, так і VSCode є кросплатформеними, тобто вони можуть працювати на різних операційних системах, таких як Windows, macOS та Linux.

5. Безкоштовність:

- Python та VSCode обидва є вільно розповсюджуваними і безкоштовними інструментами, що робить їх доступними для широкого кола розробників.

6. Інтегрована Розробка та Відлагодження:

- VSCode має вбудовану систему відлагодження, що дозволяє зручно відлагоджувати код без необхідності встановлення додаткових інструментів.
- Python також має ряд інструментів для відлагодження, які легко інтегруються з VSCode.

Вибір Python та VSCode є зручним та потужним для розробки програм, що взаємодіють з протоколом Modbus. Ці інструменти мають великий потенціал для швидкої і ефективної розробки, а також для легкої інтеграції з іншими технологіями та бібліотеками.

Для реалізації прикладу обміну даними між двома пристроями через протокол Modbus, ми можемо використовувати мову програмування Python та бібліотеку PyModbus. В даному прикладі будемо імітувати обмін даними між майстром (Master) та ведучим (Slave) пристроями. Зазначте, що це лише спрощений приклад, і у реальних умовах вам, можливо, доведеться взаємодіяти з реальними пристроями чи використовувати спеціальне обладнання.

```
1: pip install pymodbus
```

Тепер, наведемо приклад коду для обміну даними між майстром та ведучим пристроями:

Ведучий пристрій (Slave):

pip install pymodbus *from pymodbus.server.sync import StartTcpServer*

```
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer
# Задаємо адресу пристрою
slave_address = 1

# Створюємо контекст ведучого пристрою
store = ModbusSlaveContext(
    di=ModbusSequentialDataBlock.create(),
    co=ModbusSequentialDataBlock.create(),
    hr=ModbusSequentialDataBlock.create(),
    ir=ModbusSequentialDataBlock.create(),
)

context = ModbusServerContext(slaves=store, single=True)

# Запускаємо сервер Modbus на порті 5020
StartTcpServer(context, address=("localhost", 5020))
```

Майстер (Master):

```
from pymodbus.client.sync import ModbusTcpClient

# Підключаємося до ведучого пристрою (адреса та порт)
client = ModbusTcpClient("localhost", 5020)

# Адреса ведучого пристрою (слідкуйте, щоб вона відповідала адресі ведучого пристрою)
slave_address = 1

# Зчитуємо дані з ведучого пристрою (приклад для читання 5 регістрів)
result = client.read_holding_registers(0, 5, unit=slave_address)

# Виводимо результат
if result.isError():
    print(f"Помилка: {result}")
else:
    print(f"Отримані дані: {result.registers}")

# Закриваємо підключення
client.close()
```

У цьому прикладі ведучий пристрій запускає сервер Modbus на локальній машині та слухає порт 5020. Майстер встановлює з'єднання з ведучим пристроєм через той самий порт і зчитує дані з 5 регістрів. Ми використовуємо функції read_holding_registers для читання регістрів, але можна також використовувати інші функції, такі як write_registers для запису даних.

У прикладі були використані дві основні бібліотеки: pymodbus для роботи з протоколом Modbus та pymodbus.server.sync для створення ведучого пристрою (Slave), і pymodbus.client.sync для створення майстра (Master). Давайте розглянемо ці бібліотеки та інструменти більш детально:

1. pymodbus: це бібліотека Python для взаємодії з протоколом Modbus. Вона дозволяє легко реалізувати як ведучі, так і майстерні пристрої з використанням

різних варіантів транспортного рівня (TCP/IP, RTU, ASCII).

Характеристики:

Підтримка різних режимів протоколу Modbus (RTU, ASCII, TCP/IP).

Легко використовується для створення власних серверів та клієнтів Modbus.

2. pymodbus.server.sync та pymodbus.client.sync: ці модулі pymodbus використовуються для створення синхронних ведучих та майстерні пристроїв Modbus. Синхронні функції означають, що код блокується, доки не завершиться операція, що робить його простим у використанні для прикладів та демонстрацій.

Характеристики:

Синхронні функції для блокуючого обміну даними.

Забезпечують простий та інтуїтивно зрозумілий інтерфейс для реалізації ведучих та майстерні пристроїв.

3. ModbusSequentialDataBlock: клас забезпечує блочні дані для ведучого пристрою. Він використовується для імітації реєстрів ведучого пристрою та зберігання даних, які можуть бути прочитані або записані через протокол Modbus.

Характеристики:

Дозволяє легко створювати та робити доступними дані для взаємодії через протокол Modbus.

Для перевірки правильності обміну даними за допомогою протоколу Modbus можна розробити тестові сценарії, які оцінюватимуть коректність взаємодії між майстром та ведучим пристроєм. Нижче наведено приклад тестових сценаріїв на основі бібліотеки pytest, яка є популярною в тестуванні Python-програм:

1. Тест читання з регістрів:

Опис: Перевіряє, чи може майстер успішно прочитати дані з регістрів ведучого пристрою.

Тестовий сценарій:

```
    def test_read_holding_registers():
        # Підключення до ведучого пристрою
        client = ModbusTcpClient("localhost", 5020)

        # Адреса ведучого пристрою та кількість регістрів для читання
        slave_address = 1
        register_address = 0
        num_registers = 5

        # Читання з регістрів
        result = client.read_holding_registers(register_address, num_registers, unit=slave_address)

        # Перевірка, чи отримано дані без помилок
        assert not result.isError(), f"Помилка читання: {result}"

        # Перевірка, чи отримані дані вірні
        assert len(result.registers) == num_registers, "Невірна кількість отриманих регістрів"

        # Закриття підключення
        client.close()
```

2. Тест запису в регістри:

Опис: Перевіряє, чи може майстер успішно записати дані у регістри ведучого пристрою.

Тестовий сценарій:

```
    def test_read_holding_registers():
        # Підключення до ведучого пр
    def test_write_registers():
        # Підключення до ведучого пристрою
        client = ModbusTcpClient("localhost", 5020)

        # Адреса ведучого пристрою та дані для запису
        slave_address = 1
        register_address = 0
        data_to_write = [100, 200, 300, 400, 500]

        # Запис в регістри
        result = client.write_registers(register_address, data_to_write, unit=slave_address)

        # Перевірка, чи запис виконаний без помилок
        assert not result.isError(), f"Помилка запису: {result}"

        # Закриття підключення
        client.close()

```

3. Тест взаємодії з різними типами регістрів:

Опис: Перевіряє, чи майстер може взаємодіяти з різними типами регістрів, такими як входи (Input Registers), виходи (Coils), введення (Inputs), та інші.

Тестовий сценарій:

```
    def test_interaction_with_different_registers():
        # Підключення до ведучого пристрою
        client = ModbusTcpClient("localhost", 5020)

        # Адреса ведучого пристрою та кількість регістрів для взаємодії
        slave_address = 1
        num_registers = 5

        # Взаємодія з різними типами регістрів
        result_holding_registers = client.read_holding_registers(0, num_registers, unit=slave_address)
        result_input_registers = client.read_input_registers(0, num_registers, unit=slave_address)
        result_coils = client.read_coils(0, num_registers, unit=slave_address)
        result_discrete_inputs = client.read_discrete_inputs(0, num_registers, unit=slave_address)

        # Перевірка, чи отримано дані без помилок
        assert not any(result.isError() for result in [result_holding_registers, result_input_registers, result_coils, result_discrete_inputs]), "Помилка взаємодії з регістрами"

        # Закриття підключення
        client.close()
```

Проведення тестів та аналіз результатів важливі для визначення правильності роботи програми та її взаємодії з пристроями через протокол Modbus. Для цього можна використовувати фреймворк тестування, наприклад, pytest, який дозволяє легко створювати та виконувати тести у вашому проекті.

Проведення тестів з використанням pytest:

1. Встановлення pytest:

```
pip install pytest
```

2. Створення файлу з тестами, наприклад, test_modbus.py:

```
# test_modbus.py

from pymodbus.client.sync import ModbusTcpClient

def test_read_holding_registers():
    # Тест читання з регістрів
    # ...

def test_write_registers():
    # Тест запису в регістри
    # ...

def test_interaction_with_different_registers():
    # Тест взаємодії з різними типами регістрів
    # ...
```

2. Виконання тестів:

```
pytest test_modbus.py
```

Аналіз результатів:

1. Успішні тести:

- Якщо всі тести успішно пройшли, ви отримаєте вивід у вигляді "passed".
- Приклад успішного виконання тестів:

```
========== test session starts ==========
...
collected 3 items

test_modbus.py ...                                    [100%]

========= 3 passed in 0.12 seconds ==========
```

2. Помилки або невдачі тестів:

- Якщо тест не пройшов успішно, ви отримаєте повідомлення про помилку та стек виклику, які вказують на місце, де виникла проблема.

- Приклад невдачі тесту:

```
========== test session starts ==========
...
collected 3 items

test_modbus.py F..                                    [100%]

==========  FAILURES ==========
_____ test_read_holding_registers _____

>   assert not result.isError(), f"Помилка читання: {result}"
E   AssertionError: Помилка читання: Read register request failed

...

========== 1 failed, 2 passed in 0.20 seconds ==========
```

3. Логи та вивід:

- Докладніше аналізуйте логи та вивід тестів, щоб зрозуміти, де виникає проблема.

- Звертайте увагу на повідомлення про помилки та порівнюйте отримані дані із сподіваними результатами.

4. Виправлення та перевірка:

- Виправте проблеми, які були виявлені під час тестів.
- Повторно виконайте тести, щоб переконатися, що проблема вирішена.

Аналіз результатів тестів є важливим етапом розробки, оскільки він дозволяє виявити та виправити помилки перед розгортанням системи в реальних умовах.

У ході дослідження обміну даними за допомогою протоколу Modbus було встановлено та описано ключові етапи цього процесу, визначено ролі майстра та ведучого, оглянуто об'єкти даних та розглянуті різні аспекти обміну, такі як асинхронний та синхронний підходи.

Враховано можливі помилки та надано стратегії їх обробки. Обрано мову програмування Python та середовище розробки VSCode для створення прикладу обміну даними. Застосовано бібліотеку PyModbus для реалізації ведучого та майстра.

Розроблено тестові сценарії для перевірки правильності обміну даними, а проведення тестів та аналіз результатів виявили можливі помилки та дали змогу оцінити продуктивність системи.

Виокремлено питання оптимізації витрат ресурсів та швидкодії, включаючи рекомендації щодо многопоточності, кешування даних та вибір оптимальних бібліотек та фреймворків.

Загальною метою було розроблення надійного та ефективного засобу обміну даними по комп'ютерним мережам на основі протоколу Modbus.

3.2 Проблеми та виклики використання MODBUS

Modbus - це промисловий протокол зв'язку, який використовується для обміну даними між електронними пристроями в автоматизованих системах. Тут розглянемо деякі можливі проблеми та виклики, пов'язані з використанням Modbus:

Безпека:

Modbus був створений без значного фокусу на безпеці, тому це може бути потенційним викликом, особливо при використанні в мережах Інтернету.

Швидкість та пропускна здатність:

Modbus може мати обмежену пропускну здатність, що може стати проблемою в сучасних великих системах, де потрібно передавати великі обсяги даних.

Керування конфліктами із іншими пристроями:

Оскільки Modbus використовує простий Master/Slave або Client/Server підхід, можуть виникати конфлікти при спробі взаємодії більш ніж двох пристроїв одночасно.

Реалізація та інтерпретація стандартів:

Різні вендори можуть реалізувати Modbus по-різному, що може створювати проблеми сумісності між пристроями різних виробників.

Обмежені можливості адресації:

Modbus використовує 16-бітні адреси, що може обмежити кількість підключених пристроїв.

Нестабільність зв'язку:

Неправильне налаштування або перешкоди на шляху передачі даних можуть призвести до втрати з'єднання.

Відсутність механізму перевірки цілісності даних:

Modbus не має вбудованого механізму перевірки цілісності даних, що може призвести до передачі помилкових даних.

Обмежена підтримка типів даних:

Modbus має обмежену підтримку типів даних порівняно з сучасними протоколами.

Відсутність механізму синхронізації часу:

Modbus не має стандартного механізму синхронізації часу між пристроями.

Для розв'язання цих викликів можуть використовуватися додаткові технології, такі як VPN для забезпечення безпеки, або можуть використовуватися більш сучасні промислові протоколи з вдосконаленими функціональністю і безпекою.

В Україні використання протоколу Modbus може стикатися з рядом специфічних проблем та викликів, які випливають із контексту розвитку промисловості та інфраструктури в країні. Ось деякі можливі проблеми та виклики:

Старі технології:

В багатьох промислових секторах в Україні може існувати велика кількість застарілої техніки та обладнання, яке підтримує Modbus, але може бути менш сумісним з більш сучасними промисловими протоколами.

Низька швидкість Інтернету:

У деяких регіонах України може бути обмежена швидкість та доступність Інтернету, що може вплинути на здатність Modbus ефективно обмінюватися даними в режимі реального часу.

Відсутність стандартизації:

У промислових підприємствах може відсутній чіткий стандарт чи норматив, що регулює використання Modbus, що призводить до різноманітності реалізацій та проблем інтеграції.

Брак кваліфікованих кадрів:

Низький рівень освіти та брак кваліфікованих фахівців у сфері інформаційних технологій може ускладнити ефективне впровадження та підтримку систем, що використовують Modbus.

Безпека та захист від кіберзагроз:

У зв'язку зі зростанням кількості кібератак та загроз кібербезпеки, використання протоколу Modbus без належних заходів безпеки може стати джерелом загроз для промислових систем.

Обмеження доступу до технологій Інтернету речей (IoT):

Завдяки віддаленій географії та розвитку сільськогосподарської галузі, може бути складно впроваджувати сучасні IoT-рішення, що можуть використовувати інші протоколи, крім Modbus.

Потреба в модернізації:

Багато підприємств можуть стикатися з потребою в модернізації обладнання та систем, щоб вони були більш сумісними з сучасними технологіями, що може включати в себе перехід на інші протоколи.

Спроби інтеграції з іншими системами:

При спробах інтеграції Modbus з іншими системами можуть виникати проблеми через різноманітність інтерфейсів та реалізацій протоколу.

Для подолання цих викликів важливо ретельно планувати впровадження та забезпечувати якісну підтримку та навчання персоналу. Також, врахування місцевих особливостей і належне врахування кібербезпеки може сприяти успішному використанню Modbus в українських умовах.

3.3 Перспективи розвитку MODBUS

Протокол MODBUS є одним з найпоширеніших індустріальних протоколів зв'язку, використовуваних для обміну даними між електронними пристроями в автоматизованих системах. Враховуючи загальний напрямок розвитку індустріальної автоматизації і IoT, MODBUS також розвивається відповідно до нових вимог та технологій. Ось деякі потенційні перспективи розвитку MODBUS:

Вдосконалення безпеки: З урахуванням зростання загроз кібербезпеки, можна очікувати, що розвиток MODBUS буде спрямований на вдосконалення заходів безпеки, таких як шифрування даних, аутентифікація та авторизація, щоб захистити від несанкціонованого доступу та атак.

Підтримка IPv6: Зі зростанням кількості підключених пристроїв і розширенням Інтернету речей (IoT), можливо, MODBUS буде адаптований для підтримки IPv6, щоб забезпечити достатню адресацію для всіх пристроїв у мережі.

Інтеграція з хмарами: Враховуючи популярність хмарових технологій, можливо, MODBUS буде розвиватися в напрямку легшої інтеграції з хмаровими платформами, щоб забезпечити більш ефективний моніторинг та управління пристроями.

Підтримка швидших швидкостей передачі даних: З введенням нових технологій та зростанням потреб в швидкості обміну даними може з'явитися підтримка високошвидкісного обміну даними в протоколі MODBUS.

Розширення функціональності: MODBUS може розвиватися у напрямку розширення функціональних можливостей, таких як підтримка нових типів даних, покращення обробки помилок та механізмів відладки.

Стандартизація і інтероперабельність: Для забезпечення кращої взаємодії між різними пристроями та системами розробники можуть працювати над стандартизацією та підтримкою інтероперабельності.

Зазначте, що ці прогнози є загальними і можуть бути вже застарілими або доповненими відповідно до подальших розвитків в індустрії. Якщо у вас є конкретні питання про MODBUS або інші протоколи, будь ласка, уточніть їх.

В Україні Modbus також широко використовується в промисловості. Він є стандартом для зв’язку між промисловими контролерами та пристроями, такими як датчики, виконавчі механізми, реле та інші.

Перспективи розвитку Modbus в Україні є позитивними. Цей протокол має ряд переваг, які роблять його привабливим для використання в промисловості:

Відкритість: Modbus є відкритим протоколом, що означає, що його можна використовувати безкоштовно.

Простота: Modbus є простим у використанні та налаштуванні.

Ефективність: Modbus є ефективним у використанні мережевих ресурсів.

Зростання промислового виробництва в Україні стимулюватиме розвиток Modbus. Цей протокол буде використовуватися для зв’язку між новими промисловими контролерами та пристроями, що будуть встановлюватися на підприємствах.

Ось деякі конкретні приклади перспектив використання Modbus в Україні:

Впровадження розумних заводів: Modbus може використовуватися для зв’язку між різними компонентами розумного заводу, такими як сенсори, виконавчі механізми та системи управління.

Розвиток нових технологій: Modbus може використовуватися для зв’язку між новими технологіями, такими як штучний інтелект та машинне навчання.

Інтеграція з іншими системами: Modbus може використовуватися для інтеграції промислових систем з іншими системами, такими як системи управління ресурсами та системи безпеки.

Modbus є важливим протоколом для промисловості в Україні. Він має ряд переваг, які роблять його привабливим для використання в різних промислових застосуваннях. Перспективи розвитку Modbus в Україні є позитивними.

4 ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Однією з ключових аспектів, які потрібно враховувати при дослідженні обміну даними по комп'ютерним мережам на основі протоколів Modbus, є забезпечення безпеки і відповідності стандартам охорони праці та навколишнього середовища. У даному розділі буде проведено аналіз впливу використання протоколів Modbus на параметри безпеки праці та збереження екологічної стабільності.

Перший крок у забезпеченні безпеки праці - це проведення оцінки ризиків інфраструктури мережі, що використовує протоколи Modbus. Визначення потенційних загроз для працівників пов'язане з реалізацією цих протоколів, а також із врахуванням можливих витоків даних, атак та несанкціонованого доступу.

Оцінка ризиків інфраструктури мережі є критичним етапом в забезпеченні безпеки праці та навколишнього середовища при дослідженні обміну даними по комп'ютерним мережам на основі протоколів Modbus. Цей процес передбачає детальний аналіз потенційних небезпек, які можуть виникнути внаслідок використання даного протоколу.

Оцінка ризиків починається з ідентифікації потенційних загроз для безпеки інфраструктури мережі. Це можуть бути атаки ззовні, несанкціонований доступ, програмні помилки або технічні витоки. Аналіз включає також визначення вразливостей інфраструктури, які можуть бути використані для здійснення атак.

На основі ідентифікованих загроз проводиться аналіз можливих наслідків аварійного впливу на працю персоналу та екологічну безпеку. Це включає в себе оцінку ризику для здоров'я працівників та можливих негативних впливів на навколишнє середовище внаслідок витоку конфіденційної інформації чи пошкодження інфраструктури.

Після оцінки ризиків розробляються конкретні стратегії та заходи безпеки, спрямовані на зменшення впливу виявлених загроз. Це може включати в себе вдосконалення захисту мережі, встановлення систем моніторингу безпеки, регулярне оновлення програмного забезпечення та навчання персоналу з питань кібербезпеки.

Загалом, оцінка ризиків інфраструктури мережі є стратегічним етапом, спрямованим на запобігання небезпекам, забезпечення безпеки праці та мінімізацію впливу на навколишнє середовище при дослідженні обміну даними за допомогою протоколів Modbus.

На основі виявлених ризиків слід розробити та впровадити заходи безпеки, спрямовані на мінімізацію можливостей виникнення аварій, які можуть впливати на працю персоналу. Це може включати в себе регулярні оновлення програмного забезпечення, використання шифрування даних та встановлення мережевих бар'єрів.

Дослідження обміну даними по протоколах Modbus також повинно включати в себе аналіз впливу на використання ресурсів інформаційно-комунікаційних систем. Це важливо для ефективного використання енергетичних ресурсів та зниження негативного впливу на навколишнє середовище.

Розгляд ефективності використання електроенергії у контексті обміну даними по протоколах Modbus важливий для забезпечення сталого розвитку та відповідності екологічним стандартам. Впровадження енергоефективних методів комунікації може сприяти зменшенню викидів та оптимізації споживання електроенергії.

Дослідження впливу на використання ресурсів у контексті обміну даними за протоколами Modbus включає в себе глибокий аналіз ефективності інформаційно-комунікаційних систем та їхнього впливу на загальне споживання ресурсів. Аспекти такого дослідження охоплюють ефективність використання електроенергії, оптимізацію використання інших ресурсів та визначення впливу на сталість розвитку.

Важливо розглядати технологічні рішення та їхній внесок у використання електроенергії, адже ефективні методи комунікації можуть значно впливати на споживання електроенергії та зменшувати витрати. Крім того, оптимізація ресурсного використання дозволяє забезпечити сталість розвитку інформаційних систем, уникнути перенавантаження та зменшити витрати на обслуговування.

Аналіз впливу на використання ресурсів є ключовим етапом для розуміння та управління сталим розвитком технологій обміну даними за протоколами Modbus.

Розділ "Охорона праці та навколишнього середовища" в контексті дослідження обміну даними по комп'ютерним мережам на основі протоколів Modbus є необхідним для повноцінного розуміння імплікацій використання цих протоколів. Забезпечення безпеки праці та врахування впливу на навколишнє середовище є ключовими компонентами успішної імплементації технологій обміну даними для сталого розвитку та збереження природних ресурсів.

5 ЕКОНОМІЧНА ЧАСТИНА

У сучасному світі, де технології відіграють ключову роль у розвитку різних галузей промисловості, ефективний обмін даними в комп'ютерних мережах стає важливою передумовою для забезпечення стабільності та продуктивності підприємств. Особливо актуальною є проблема взаємодії технічних пристроїв, що використовують різні протоколи для обміну інформацією. У даному дослідженні зосереджено увагу на аналізі та оптимізації обміну даними на основі протоколів Modbus, які широко використовуються у промислових системах та автоматизації.

Одним із ключових аспектів, які вивчаються у магістерській роботі, є економічний вплив впровадження протоколів Modbus у виробничому середовищі. Зокрема, досліджується вартість впровадження нових технологій, адаптація існуючих пристроїв до нових стандартів, а також ефективність використання ресурсів при здійсненні переходу на дані протоколи.

Аналізуються витрати на навчання персоналу, встановлення та підтримання нових обладнань та програмного забезпечення. Визначається очікуваний економічний приріст, пов'язаний із застосуванням протоколів Modbus, у вигляді покращення швидкості обміну даними, зменшення часу простою обладнання та підвищення загальної ефективності виробничого процесу.

Робота також включає вивчення досвіду впровадження аналогічних технологій в інших компаніях та галузях, аналізуючи їхні позитивні та негативні результати. Це дозволяє визначити передові практики та уникнути можливих помилок під час впровадження на власному підприємстві.

У підсумку, економічний аспект дослідження є ключовим для визначення практичної цілеспрямованості та доцільності впровадження протоколів Modbus в конкретному виробничому середовищі. Враховуючи витрати та очікувані вигоди, дослідження спрямоване на розкриття економічного потенціалу інтеграції цих протоколів у системи обміну даними, що виходять за межі технічних аспектів, і визначає реальні переваги для підприємства в цілому.

ЗАГАЛЬНИЙ ВИСНОВОК

У результаті вивчення протоколів зв'язку та аналізу їхніх застосувань, в даній роботі детально розглянуто протокол MODBUS як ефективний і широко використовуваний засіб обміну даними в індустріальних системах.

У розділі 1 визначено загальні принципи протоколів зв'язку, а також розглянуто вступні аспекти MODBUS, що створило необхідну основу для подальшого розгляду його реалізації.

Розділ 2 детально описує механізми та засоби реалізації протоколу MODBUS на різних рівнях його стеку, включаючи фізичний, канальний та мережевий рівні. Також вивчено засоби реалізації для різних платформ, що підкреслює універсальність та адаптивність протоколу.

У розділі 3 проаналізовано практичні застосування протоколу MODBUS в промисловості, висвітлено існуючі проблеми та виклики його використання. Також розглянуто перспективи розвитку MODBUS, враховуючи сучасні вимоги та тенденції в індустріальних системах.

Висновки роботи відображають ключові аспекти, виявлені під час дослідження. MODBUS, як протокол зв'язку, виявився важливим інструментом для забезпечення ефективного обміну даними в промислових системах. Однак існують виклики, які потребують уважної уваги, такі як забезпечення безпеки та відмові стійкості.

Робота також вказує на необхідність подальших досліджень та розвитку протоколу MODBUS для адаптації до зростаючих вимог індустрії. При цьому важливо приділяти увагу стандартизації та вдосконаленню засобів захисту для забезпечення надійності та безпеки обміну даними в промислових середовищах.

Загальною висновком можна сказати, що робота дозволила отримати глибоке розуміння протоколу MODBUS та його застосувань, визначити переваги та обмеження, а також вказати напрямки для подальших досліджень у цьому напрямку.

Завершуючи, робота вважається важливим внеском у розуміння протоколу MODBUS та його впливу на промислові системи. Зазначена структурованість та комплексність розгляду відображається у всіх аспектах протоколу, включаючи фізичні та мережеві рівні, а також практичні аспекти його застосування.

Незважаючи на свою успішну історію, MODBUS стикається з викликами, такими як забезпечення безпеки та відмові стійкості в умовах сучасних промислових середовищ. Застосування протоколу вимагає постійного вдосконалення та адаптації до змінних умов ринку та технологічних вимог.

У висновках роботи також акцентується важливість подальших досліджень і розвитку, спрямованих на оптимізацію MODBUS та врахування сучасних тенденцій, таких як Інтернет речей (IoT) та штучний інтелект. Ці нові виклики визначають напрямок для майбутнього розвитку протоколу, який має потенціал для подальшого збільшення його ефективності та ширшого застосування в індустріальних та автоматизованих системах.

Загальна висновок роботи вказує на актуальність та значущість вивчення протоколів зв'язку, зокрема MODBUS, що відкриває нові можливості для подальших досліджень та вдосконалення промислових систем у майбутньому.

ПЕРЕЛІК ПОСИЛАНЬ

1) Повний посібник з протоколу серійного зв'язку та нашого RS-485. 2019 р. Доступно за посиланням: https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3884.html

2) Modbus FAQ: Про Організацію Modbus. 2005 р. Доступно за посиланням: https://www.Modbus.org/faq.php

3) Специфікація протоколу застосування Modbus V1.1b3. 2012 р. Доступно за посиланням: https://Modbus.org/specs.php

4) Сторінка з описом продукту Spice IoT-TICKET. 2019 р. Доступно за посиланням: https://www.wapice.com/products/iot-ticket

5) Технічний опис до ардуіно Arduino Uno. 2013 р. Доступно за посиланням: https://www.farnell.com/datasheets/1682209.pdf

6) Технічний опис до ATmega328P. 2015 р. Доступно за посиланням: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

7) Технічний опис до ATmega2560. 2014 р. Доступно за посиланням: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

8) Технічний опис до MAX485ESA. 2014 р. Доступно за посиланням: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

9) Підключення мереж RS485. 2020 р. Доступно за посиланням: https://www.se.com/ww/en/faqs/FA221785/

10) Основи RS-485: Коли необхідна термінація і як її правильно виконати. 2021 р. Доступно за посиланням: https://e2e.ti.com/blogs_/b/analogwire/posts/rs-485-basics-when-termination-is-necessary-and-how-to-do-it-properly

11) Thomas Liu. 2011 р. Цифровий вихідний датчик відносної вологості та температури DHT22. Доступно за посиланням: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

12) Технічний опис до KY-009. 2017 р. Доступно за посиланням: https://datasheetspdf.com/datasheet/KY-009.html

13) Modbus-Master-Slave-for-Arduino. 2014 р. Доступно за посиланням: https://github.com/smarmengol/Modbus-Master-Slave-for-Arduino

14) SimpleDHT. 2018 р. Доступно за посиланням: https://github.com/winlinvip/SimpleDHT

15) Документація з Python 3. 2018 р. Доступно за посиланням: https://docs.python.org/3/

16) MinimalModbus. 2019 р. Доступно за посиланням: https://minimalModbus.readthedocs.io/en/stable/

17) IoTTicket-PythonLibrary. 2016 р. Доступно за посиланням: https://github.com/IoT-Ticket/IoTTicket-PythonLibrary

18) Технічний опис до чіпа CH340 для USB до послідовного зв'язку. 2017 р. Доступно за посиланням: https://cdn.sparkfun.com/datasheets/Dev/Arduino/Other/CH340DS1.PDF

19) Організація M, "Специфікація протоколу застосування Modbus v1.1b3." http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf, 2012.

20) H. I. Networks, "Modbus rtu slave interface." http://www.anybus.com/products/products.asp?PID=78&ProductType=Anybus-S.

21) Advantech, "Adam-4572." http://www2.advantech.eu/products/GF-5TZ5/ADAM-4572/mod_1E01192D-95A1-42A4-B199-79343134F4CA.aspx.

22) E. . E. E. GES.M.B.H. http://www.epluse.com/.

23) E. . E. E. GES.M.B.H, "Low power oem humidity and temperature probe with modbus interface." http://www.epluse.com/fileadmin/data/product/ee071/Datasheet_EE071.pdf.

24) M. T. Inc., "Low-power design guide." http://ww1.microchip.com/downloads/en/AppNotes/01416a.pdf, 2011.

25) S. L. Inc. http://www.silabs.com/pages/default.aspx.

26) S. L. Inc, "Efm32zg108 datasheet, rev 1.10." http://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32ZG108.pdf, 2015.

27) M. Organization. http://www.modbus.org/.

28) T. Instruments, "Interface circuits for tia/eia-rs485 (rs-485)." http://www.ti.com/lit/an/slla036d/slla036d.pdf, 2008.

29) C. Controls, "Understanding eia-485 networks." http://www.ccontrols.com/pdf/ExtV1N1.pdf, 1999.

30) P. . P. International. http://www.profibus.com/.

31) T. Electronics, "Inductive filtering modular rj jacks." http://datasheet.octopart.com/6-6609208-0-Corcom-datasheet-501201.pdf.

32) HRS, "Rj45 modular jack connectors with pulse transformers." https://www.hirose.co.jp/cataloge_hp/e22229322.pdf.

33) D.-F. T. AB, "Engineering kit data sheet." http://www.d-flow.se/sites/default/files/media/engineering_kit_data_sheet.pdf, 2015.

34) D.-F. T. AB, "Dn20 evaluation meter data sheet." http://www.d-flow.se/sites/default/files/media/dn20_evaluation_meter_data_sheet.pdf, 2015.

35) D.-F. T. AB, "D-flow ufo2 asic brief data sheet." http://www.d-flow.se/sites/default/files/media/UFO2_ASIC_brief_data_sheet.pdf, 2015.

36) M. Integrated. http://www.maximintegrated.com/en.html.

37) M. Integrated, "Datasheet max3485." http://www.intersil.com/content/dam/Intersil/documents/isl3/isl32600e-01e-02e-03e.pdf.

38) S. L. Inc. http://www.silabs.com/products/mcu/Pages/simplicity-studio.aspx.

39) R. Kwiecien, "Radzio! modbus master simulator," http://en.radzio.dxp.pl.

40) F. Corporation. http://www.fluke.com/fluke/sesv/digitala-multimetrar/advanced-multimeters/fluke-289.htm?pid=56061.

41) J. W. Eaton. http://www.gnu.org/software/octave/index.html.

42) SAFT. http://www.saftbatteries.com/.

43) SAFT, "Ls14500." http://www.saftbatteries.com/force_download/LS14500.pdf.

44) SAFT, "Ls17500." http://www.saftbatteries.com/force_download/LS17500_EN_31029_2_0615.pdf.

45) SAFT, "Ls26500." www.saftbatteries.com/force_download/LS26500.pdf.