Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут Електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра електропривода
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

(бакалавра, спеціаліста, магістра)

студента Івченка Михайла Максимовича
(ПІБ)

академічної групи 141м-17-4
(шифр)

спеціальності 141 Електроенергетика, електротехніка та електромеханіка
(код і назва спеціальності)

спеціалізації[1] _____

за освітньо-професійною програмою Електромеханічні системи автоматизації та електропривод

(офіційна назва)

на тему Система моніторингу цифрових електроприводів
_____
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	Казачковський М.М.			
розділів:				
Завдання на розробку	Казачковський М.М.			
Обґрунтування архітектурних рішень	Казачковський М.М.			
Розробка системи моніторингу…	Казачковський М.М.			
Техніко-економічне обґрунтування	Тимошенко Л.В.			
Рецензент	Галушко О.М.			
Нормоконтролер	Казачковський М.М.			

Дніпро
2018

ЗАТВЕРДЖЕНО:
завідувач кафедри
електропривода
(повна назва)

_____ Казачковський М.М.
(підпис) (прізвище, ініціали)

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, спеціаліста, магістра)

студенту Івченко М.М **академічної групи** 141м-17-4
(прізвище та ініціали) (шифр)
спеціальності 141 Електроенергетика, електротехніка та електромеханіка

спеціалізації[1] _____
за освітньо-професійною програмою Електромеханічні системи автоматизації та електропривод
(офіційна назва)
на тему Система моніторингу цифрових електроприводів._____

_____,

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____№_____

Розділ	Зміст	Термін виконання
Завдання на розробку	Вимоги до програмного забезпечення	01.10.2018-03.10.2018
Обґрунтування архітектурних рішень	Вибір протоколу передачі даних, архітектури програмного забезпечення, способу виводу інформації	05.10.2018-15.10.2018
Розробка системи моніторингу цифрових електроприводів	Опис цільового пристрою та його налаштування, алгоритму роботи програмного забезпечення на прикладі основних блоків коду.	15.10.2018-01.12.2018
Техніко-економічне обґрунтування	Визначення працеємкості опрацювання програмного продукту. Розрахунок витрат на створення програмного продукту.	02.12.2018-05.12.2018

Завдання видано _____ Казачковський М.М.
(підпис керівника) (прізвище, ініціали)

Дата видачі 15 жовтня 2018

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Івченко М.М.
(підпис студента) (прізвище, ініціали)

Реферат

Пояснювальна записка складається з 54с., 22 рис., 9 табл.

Мета роботи: розробка системи моніторингу параметрів цифрового електропривода у реальному часі.

Вибрані протокол передачі даних, архітектура додатку, мова програмування, спосіб виводу інформації. Приведені роз'яснення щодо моделі даних та доступу до них у вибраній мережі, проблеми багатопоточності веб-додатку та шляхи вирішення.

Вибраний об'єкт дослідження (перетворювач частоти Altivar Process). Розроблено веб-додаток. Приведені алгоритм роботи програмного забезпечення, структура проекту, основні можливості та функціональність головних блоків коду. Приведені можливості зображення залежності $y = f(t)$, збереження даних до файлу.

Розраховані працеємкість опрацювання програмного продукту, витрати на створення програмного продукту.

ЦИФРОВИЙ ЕЛЕКТРОПРИВОД, ПЕРЕТВОРЮВАЧ ЧАСТОТИ, АВТОМАТИЗАЦІЯ, СИСТЕМИ МОНІТОРИНГУ, SCADA, MODBUS, TCP/IP, КЛІЄНТ-СЕРВЕР, КОНТРОЛЕР, РЕГІСТРИ ПАМ'ЯТІ.

Abstract

The explanatory note includes 55 p., 22 fig., 9 tables.

Purpose: development of a system for monitoring the parameters of a digital electric drive in real time

Selected data transfer protocol, application architecture, programming language, method of output information. Explained about the data model and access to it in the selected network, the problem of multi-threading the web application and solutions.

Selected monitoring object (frequency converter Altivar Process). Web application developed. The algorithm of the software work, the structure of the project, the main features and the functionality of the main blocks of the code are given. The possibilities of the image of the dependence $y = f(t)$ are presented, saving of data to the file.

Calculated labor productivity of the software product, the cost of creating a software product.

DIGITAL ELECTRIC DRIVE, AUTOMATION, FREQUENCY CONVERTER, MONITORING SYSTEMS, SCADA, MODBUS, TCP/IP, CLIENT-SERVER, CONTROLLER, REGISTERS.

Зміст

Вступ

Збір даних на сьогодні є основним та найбільш перспективним методом автоматизованого відстеження складних динамічних систем, в критичних з точки зори безпеки та надійності областях. Саме на принципах диспетчерського керування будуються великі автоматизовані системи в промисловості, енергетиці, транспорті, космічній та воєнній галузі.

Сучасна АСУТП (автоматизована система управління технологічним процесом) являє собою багаторівневу людино-машинну систему управління. Створення системи відстеження складних технологічних процесів здійснюється з використанням автоматичних інформаційних систем збору даних і обчислювальних комплексів, які постійно вдосконалюються по мірі еволюції технічних засобів і програмного забезпечення.

Спектр функціональних можливостей визначено самої роллю програмного додатка в системах управління і реалізований практично у всіх пакетах:

- засоби виконання прикладних програм;
- збір первинної інформації від пристроїв нижнього рівня;
- обробка первинної інформації;
- реєстрація помилок;
- зберігання інформації з можливістю її пост-обробки (як правило, реалізується через інтерфейси до найбільш популярним баз даних);
- візуалізація інформації у вигляді мнемосхем, графіків, тощо;
- можливість роботи прикладної системи з наборами параметрів, що розглядаються як "єдине ціле" ("recipe" або "установки").

Розглядаючи узагальнену структуру систем управління, слід ввести і ще одне поняття - Micro-SCADA. Micro-SCADA - це системи, що реалізують стандартні (базові) функції, властиві SCADA - системам верхнього рівня, але орієнтовані на рішення задач автоматизації в певній галузі (вузькоспеціалізовані). На противагу їм SCADA - системи верхнього рівня є універсальними.

Всі компоненти системи управління об'єднані між собою каналами зв'язку. Забезпечення взаємодії SCADA - систем з локальними контролерами, контролерами верхнього рівня, офісними та промисловими мережами покладено на так зване комунікаційне ПО. Це досить широкий клас програмного забезпечення, вибір якого для конкретної системи управління визначається багатьма факторами, в тому числі і типом застосовуваних контролерів, і використовуваної SCADA - системою.

1. ЗАВДАННЯ НА РОЗРОБКУ

Метою магістерської роботи є розробка системи моніторингу параметрів цифрового електропривода у реальному часі. Результати роботи можуть бути використані у таких галузях як:

1. перетворювачі частоти та цифрові електроприводи постійного струму;
2. системи автоматизації технологічних процесів;
3. системи «розумний дім»;
4. навчальний процес.

1.1 Головні вимоги до системи

Система має відповідати наступним вимогам:

1. з’єднання системи та предмету відстеження відбувається за протоколом Modbus;
2. можливість зчитувати регістри пристрою;
3. зчитування дискретних, аналогових сигналів, а також «слово стану»;
4. можливість відстеження помилок та подальше їх збереження у базу даних або файл.

1.2 Функціональна схема

Функціональна схема взаємодії розроблюваної системи з об’єктом моніторингу зображена на рис. 1.1

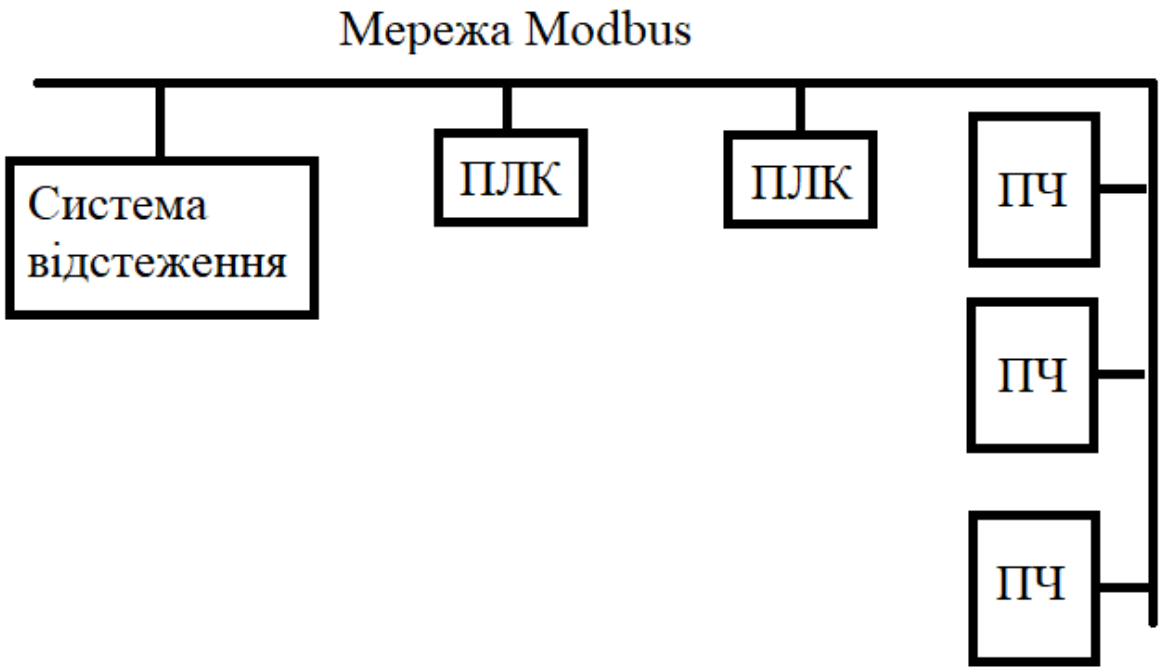

Рис.1.1 Функціональна схема взаємодії системи та відстежуваних пристроїв.

1.3 Можливі контрольовані параметри

Перелік можливих контрольованих змінних наведено у табл.1.1.

Таблиця 1.1 Можливі параметри для роботи електроприводу з системою відстеження.

Категорія	Тип
Параметри керування	WORD (BitString 16), WORD (Enumeration)
Параметри стану	WORD (BitString 16), WORD (Enumeration), UINT, INT
Параметри Входів/Виходів	WORD (BitString 16), WORD (Enumeration), UINT, INT

У табл.1.1:

WORD (BitString 16) – набір з 16 байтів кожен з яких має два стани – активний та неактивний, що вказує на певний стан системи.

WORD (Enumeration) – набір байтів змінної довжини. Число переведене з двійкової системи числення – дає певний стан цільового пристрою.

UINT – Число в діапазоні [-32767…32767]

INT – Число в діапазоні [0…32767]

1.4 Додаткові вимоги

Вимоги до надійності системи:

Надійне (стійке) функціонування системи має бути забезпечено виконанням сукупності організаційно-технічних заходів, перелік яких наведено нижче:

1. організацією безперебійного живлення технічних засобів;

2. використанням ліцензійного програмного забезпечення;

3. регулярним виконанням вимог ГОСТ 51188-98. Захист інформації. Випробування програмних засобів на наявність комп'ютерних вірусів;

4. дотриманням інструкцій по експлуатації;

5. своєчасним сервісним обслуговуванням

Вимоги до ергономіки та технічної естетики:

• система повинна бути незалежна від мовних налаштувань ПК;

• незалежність від типу та призначення цільового пристрою.

Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання:

• вимоги відповідають умовам експлуатації, технічного обслуговування, ремонту і зберігання сучасних комп'ютерів та іншого електронного обладнання.

Вимоги до захисту від зовнішніх впливів:

• відповідають сучасним вимогам до радіоелектронного захисту, стійкості до зовнішніх впливів електронного обладнання.

2. ОБҐРУНТУВАННЯ АРХІТЕКТУРНИХ РІШЕНЬ

Згідно завдання система має працювати у мережі Modbus.

2.1 Вибір протоколу

Modbus - відкритий комунікаційний протокол, заснований на архітектурі провідний-ведений (master-slave). Широко застосовується в промисловості для організації зв'язку між електронними пристроями. Може використовуватися для передачі даних через послідовні лінії зв'язку RS-485, RS-422, RS-232 і мережі TCP / IP (Modbus TCP).

2.1.1 Протокол інформаційного обміну MODBUS-RTU

При використанні RTU - режиму кожен байт повідомлення містить два чотирьохбітних шістнадцяткових числа. Кожне повідомлення передається безперервним потоком.

Формат кожного байта в RTU-режимі:

Система кодування:

- 8-ми бітна двійкова
- шістнадцяткова 0 - 9, A - F

Два шістнадцяткових числа містяться в кожному 8-ми бітному байті повідомлення.

Призначення біт:

- 1 стартовий біт
- 8 біт даних, молодшим значущим розрядом вперед
- 1 біт паритету/немає біта паритету
- 1 стоповий біт, якщо є паритет; 2 стопових біта, якщо немає паритету

Зміст MODBUS повідомлення

У RTU- режимі повідомлення починається з інтервалу тиші рівного часу передачі 3.5 символів при даній швидкості передачі в мережі. Першим полем передається адреса пристрою. Слідом за останнім переданим символом також слідує інтервал тиші тривалістю не менше 3.5 символів. Нове повідомлення повинно починатися не раніше цього інтервалу. Таким чином, якщо нове повідомлення почнеться раніше інтервалу тривалістю 3.5 символу, приймаючий пристрій сприйме його як продовження попереднього

повідомлення. В цьому випадку встановлюється помилка, оскільки виникає розбіжність контрольних сум. Типовий приклад повідомлення показаний в таблиці 2.1.

Таблиця 2.1 Структура повідомлення

Старт	Адреса	Функція	Дані	CRC	Кінець
T1-T2-T3-T4	8 біт	8 біт	N * 8 біт	16 біт	T1-T2-T3-T4

Зміст адресного поля

Адресне поле фрейму містить 8 біт. Допустима адреса передачі знаходиться в діапазоні 0 - 247. Кожному підлеглому пристрою присвоюється адреса в межах від 1 до 247. Адреса 0 використовується для широкомовної передачі, його розпізнає кожен пристрій.

Коли MODBUS протокол використовується на більш високому рівні мережі, широкомовна передача може не підтримуватися або може бути реалізована іншими методами.

Недоліки

1. Можливість помилок при десинхронізації запитів.
2. Необхідність використання додаткового пристрою для підключення ПК до мережі.

Використання даного протоколу накладає певні обмеження на підключення системи відстеження до конкретного пристрою та до мережі, оскільки вимагає придбання додаткового обладнання. Також обмеження поширюються і на розробку програмного забезпечення для опитування перетворювача частоти або будь-якого іншого пристрою.

На заміну стандарту RTU пропонується використати Modbus TCP, що дає більш гнучкий спосіб налаштування та простоту підключення.

2.1.2 Протокол інформаційного обміну MODBUS TCP/IP

Протокол MODBUS TCP / IP забезпечує обмін між пристроями в мережі Ethernet, використовуючи модель Клієнт - Сервер зі стеком протоколів TCP / IP, де перетворювач виступає в ролі MODBUS Server.

Модель Клієнт - Сервер ґрунтується на чотирьох типах повідомлень:

- MODBUS Request;
- MODBUS Confirmation;
- MODBUS Indication;
- MODBUS Response,

де MODBUS Request - запит даних з сервера; MODBUS Confirmation – повідомлення, що підтверджує прийом клієнтом відповіді від сервера; MODBUS Indication – повідомлення, що підтверджує що запит клієнта прийнятий сервером; MODBUS Response - відповідь сервера.

Модель Клієнт - Сервер зображена на рисунку 2.1.

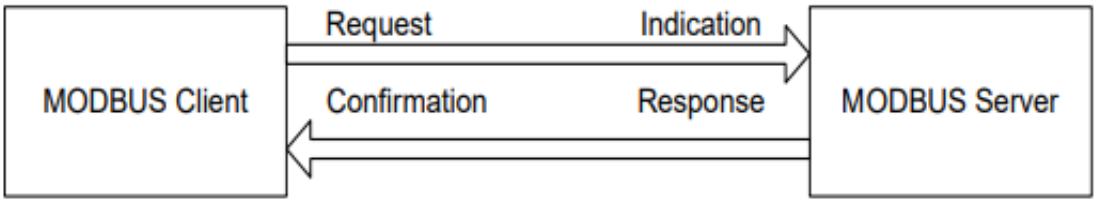

Рисунок 2.1 Модель клієнт – сервер

Дані прикладного рівня MODBUS TCP / IP

Структура пакета MODBUS TCP/IP ADU представлена на рисунку 2.2

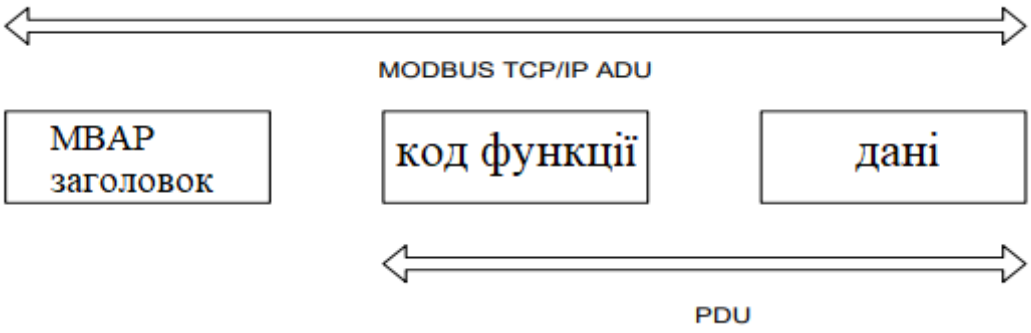

Рисунок 2.2 Структура пакета Modbus TCP

ADU для протоколу Modbus TCP наведений в таблиці 2.2

Таблиця 2.2 ADU протоколу Modbus TCP.

ID транзакції	Id протоколу	Довжина	Адреса серверу	Код функції	Дані

MBAP заголовок

MBAP (MODBUS Application Protocol) заголовок (табл. 2.3) призначений для визначення в стеці протоколів TCP / IP протоколу MODBUS TCP / IP.

Призначення полів MBAP заголовку наведено в таблиці 2.3

Таблиця 2.3 Призначення полів MBAP заголовку

Поле	Довжина	Опис	Клієнт	Сервер
Transaction Identifier (Ідентифікатор транзакції)	2	ідентифікація MODBUS Запиту/відповіді	визначається клієнтом	Встановлюється з отриманого запиту
Protocol Identifier (Ідентифікатор протоколу)	2	0 = протокол Modbus TCP	визначається клієнтом	Встановлюється з отриманого запиту
Length (Довжина)	2	Кількість байт	Визначається клієнтом (Запит)	Визначається клієнтом (Відповідь)
Unit Identifier (Ідентифікатор пристрою)	1	Ідентифікатор пристрою підключеного до мережі		

У табл. 2.3:

- Transaction Identifier - порядковий номер запиту Клієнта;

- Protocol Identifier - використовується для ідентифікації MODBUS протоколу в різних мережах;

- Length - лічильник байт переданого пакета починаючи з поля Unit Identifier;

- Unit Identifier - ідентифікатор запитуваного пристрої використовується в послідовному каналі MODBUS. Дане поле застосовується для доступу до пристроїв підключених через шлюз Ethernet / послідовний канал MODBUS.

2.2 Модель даних

MODBUS надає 4 типи даних:

- Discrete Inputs - однобітовий тип, доступний тільки для читання;

- Coils - однобітовий тип, доступний для читання і запису;
- Input Registers - 16-бітовий знаковий або беззнаковий тип, доступний тільки для читання;
- Holding Registers - 16-бітовий знаковий або беззнаковий тип, доступний для читання і запису.

Для реалізації проекту потрібне постійне опитування серверу. Специфікація протоколу визначає чотири таблиці даних (табл. 2.4).

Таблиця 2.4 Таблиці даних

Таблиця	Тип елементу	Тип доступу
Регістри прапорів	Один біт	R/W
Дискретні входи	Один біт	R
Регістри зберігання	16-бітове слово	R/W
Регістри вводу	16-бітове слово	R

У табл. 2.4: R – зчитування; W – запис.

Доступ до елементів в кожній таблиці здійснюється за допомогою 16-бітної адреси, першій клітинці відповідає адреса 0. Таким чином, кожна таблиця може містити до 65536 елементів. Специфікація не визначає, що фізично повинні представляти собою елементи таблиць і по яким внутрішнім адресами пристрою вони повинні бути доступні.

2.3 Доступ до даних

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

- 1 (0x01) - читання значень з декількох регістрів прапорів (Read Coil Status);
- 2 (0x02) - читання значень з декількох дискретних входів (Read Discrete Inputs);
- 3 (0x03) - читання значень з декількох регістрів зберігання (Read Holding Registers);
- 4 (0x04) - читання значень з декількох регістрів вводу (Read Input Registers).

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

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

Значення регістрів зберігання і регістрів введення передаються починаючи із зазначеної адреси, по два байта на регістр, старший байт кожного регістра передається першим. Розподілення байтів наведено в таблиці 2.5.

Таблиця 2.5 Розподілення байтів при зчитуванні регістрів

Байт 1	Байт 2	Байт 3	Байт 4	….	Байт 6	Байт 7
$R_{A,1}$	$R_{A,0}$	$R_{A+1,1}$	$R_{A+1,0}$	….	$R_{A+Q-1,1}$	$R_{A+Q-1,0}$

Значення прапорів і дискретних входів передаються в упакованому вигляді - по одному біту на прапор. Одиниця означає включений стан, нуль - вимкнений. Значення запитаних прапорів заповнюють спочатку перший байт, починаючи з молодшого біта, потім наступні байти, також від молодшого біта до старших. Молодший біт першого байта даних містить значення прапора, зазначеного в полі «адреса». Якщо запитана кількість прапорів не кратна восьми, то значення зайвих бітів заповнюються нулями. Розподілення байтів при читанні прапорів наведено у таблиці 2.6.

Таблиця 2.6 Розподілення байтів при зчитуванні прапорів

Байт 1								…	Байт N					
F_{A+7}	F_{A+7}	F_{A+7}	F_{A+7}	F_{A+7}	F_{A+7}	F_{A+7}	F_A	…	0	…	0	F_{A+Q-1}	F_{A+Q-2}	…

2.4 Слово стану

Слово стану являє собою набір з 16 біт, кожен з яких відображує певну характеристику і може приймати два логічних значення – 0 та 1. Так наприклад слово стану для перетворювача частоти ATV 900 при роздільному/спільному каналі керування наведено в таблиці 2.7.

Таблиця 2.7 Слово стану ATV 900 при роздільному/спільному каналі керування

Біт	Опис, значення
0	«ПЧ готовий до вмикання», чекає вмикання силового живлення
1	«Вмикання», ПЧ готовий
2	«Робота активована», робота
3	Стан знайденої помилки функціонування: 0 - неактивно 1 - активно
4	«Живлення є», силове живлення підключено 0 – силове живлення є 1 – силового живлення немає
5	Швидка зупинка
6	«Живлення знято», силове живлення вимкнене
7	Попередження: 0 – немає попереджень 1 – є попередження
8	Зарезервований (=0)
9	Дистанційне керування: керування чи завдання по мережі 0 – керування чи завдання з графічного терміналу 1 – керування чи завдання по мережі
10	Необхідне завдання досягнуте: 0 – не досягнуте 1 – досягнуте
11	«Внутрішнє обмеження активне», завдання поза обмежень 0 – завдання всередині обмежень 1 – завдання поза обмеженнями
12	Зарезервований
13	Зарезервований
14	«Кнопка STOP», зупинка за допомогою кнопки STOP 0 – не натиснута 1 – натиснута
15	«Напрямок», напрямок обертання 0 – вперед 1 – назад

На запит слова стану до перетворювача частоти або будь-якого іншого пристрою прийде відповідь у шістнадцятковому або двійковому форматі, що буде являти собою стан пристрою, наприклад:

Response = 647H = 1607D = 0110 0100 0111

(максимальне число, яке можливо отримати, дорівнює у десятковому форматі 32767, у двійковому 0111 1111 1111 1111)

Як видно з прикладу відповідь являє собою набір з нулів та одиниць, при цьому немає проблеми в тому, щоб розібрати кожну з них та поставити з необхідною змінною, але існує проблема в тому, що слово стану змінює набір параметрів. Так, для того ж перетворювача частоти при зміні каналу керування на I/O, ми отримаємо зовсім інший набір характеристик, не кажучи вже про підключення до іншого пристрою.

Виходячи з описаного вище, інформацію про стан предмету відстеження логічніше та практичніше виводити як число або набір бітів.

2.5 Вибір мови програмування

На сьогоднішній день існує велика кількість мов для написання програм, які, так чи інакше, підходять для реалізації проекту. Так можливо виділити основні:

1. C/C++;
2. Assembler;
3. Java/C#/Python.

Такі мови, як C/C++ та Assembler дуже швидко компілюються та працюють, але мають великий недолік – не орієновані на вивід інформації до клієнта (вікно, веб-сторінка, мобільний пристрій та інше).

Такі мови, як Java/C#/Python дуже схожі між собою, оскільки всі є об’єктно-орієнтованими та виконують майже одні й ті ж самі функції. Було прийнято рішення обрати мову Java, з наступних причин:

1. незалежність від платформи;
2. величезна кількість бібліотек написаних сторонніми розробниками;
3. найширша спільнота розробників.

2.6 Вивід інформації

На даному етапі спроектована система (рис. 2.3), що зв’язується з необхідним пристроєм по мережі Modbus TCP.

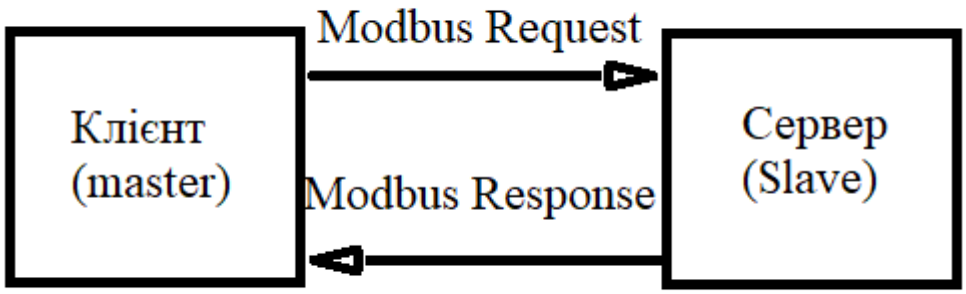

Рисунок 2.3 – Функціональна схема проекту

Недоліком є те, що на даному етапі виконувати роль опитувача буде написана та незмінна програма на мові Java. Вивід даних можливо реалізувати тільки у консоль, що не є сприятливим для користувача інтерфейсом.

Вивід інформації, що буде зрозумілою для користувача, можливо реалізувати, змінивши роль клієнта на роль проміжного сервера, що буде обробляти запити, які надходять ззовні, формувати та надсилати необхідні команди до цільового пристрою, та згідно його відповідей формувати дані, що будуть сприятливими для людини. Реалізувати дану систему можливо як віконну програму або веб сервіс.

Недоліком віконної програми є прив’язка до конкретного пристрою, на якому потрібно інсталювати програмні компоненти для компіляції та запуску програми. Натомість, реалізувавши систему як веб сервіс, підключитись до нього буде можливим з будь-якого пристрою, що має вихід у інтернет та браузер (ПК, телефон, планшет), що знаходиться у тій самій мережі, що значно розширює функціонал системи.

Функціональна схема приведена на рисунку 2.4

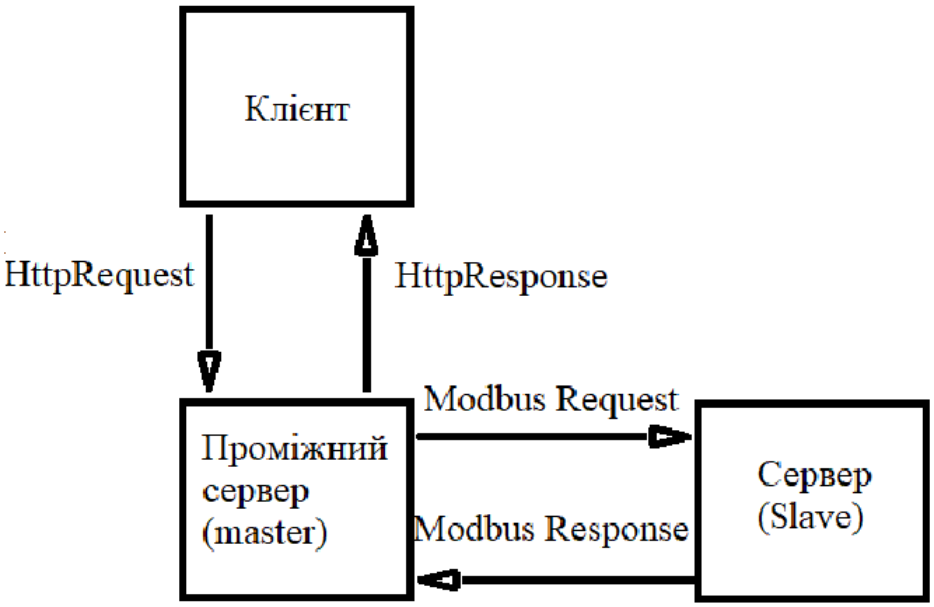

Рисунок 2.4 Функціональна схема веб сервісу

2.7 Принцип роботи кліент + сервер (Java Servlets)

Сервлет - програма Java, що запускається і виконується в контейнері сервера програм. Клієнт спілкується з такою програмою за допомогою веб-браузера (Рисунок 2.5)

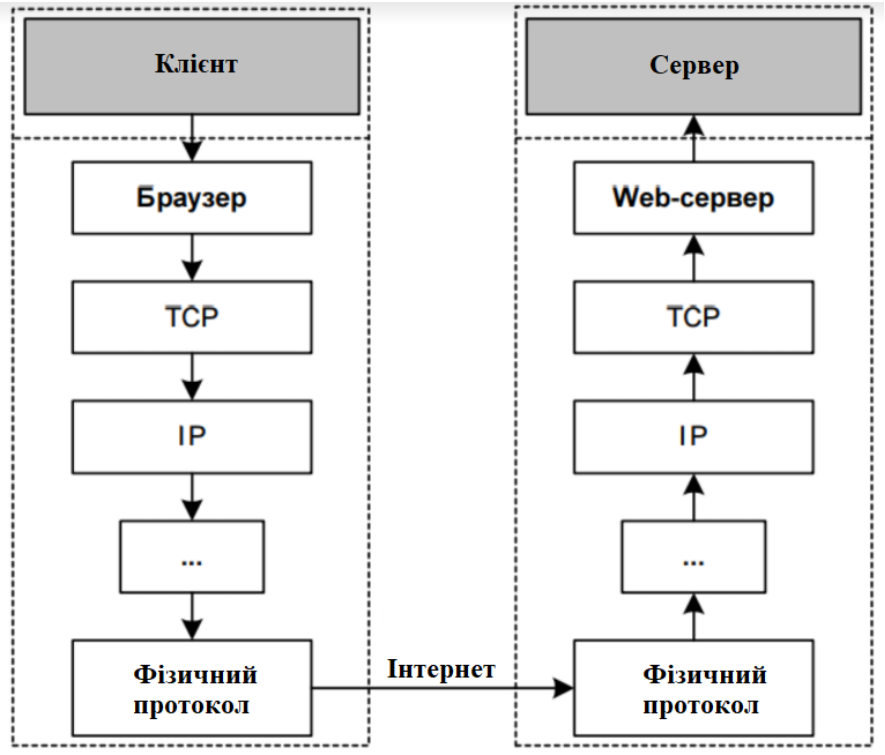

Рисунок 2.5 Клієнт + сервер у мережі інтернет

Ніяких додаткових програм на стороні клієнта встановлювати не потрібно. Сервлети підтримуються віртуальною машиною JVM, що запобігає витоку пам'яті і забезпечує функціонування garbage collection.

Кожному клієнту сервлет виділяє незалежний потік виконання. Клієнт посилає застосунком HTTP-запит, сервлет генерує відповідь і повертає його клієнту у вигляді html-документу.

Сервлет має наступні характеристики:

- компонент додатків Java Enterprise Edition;
- загружается веб-сервером в контейнер;
- виконуєтьсяся на стороні сервера;
- обробляє клієнтські запити;
- динамічно генерує відповіді на запити;
- знаходиться у стані очікування, якщо запити відсутні;
- приймає запити від інших сервлетів (Servlet chaining);
- підтримує з'єднання з ресурсами.

Найбільшого поширення набули сервлети, що обробляють клієнтські запити по протоколу HTTP. Технологія сервлетів є оболочкою протоколу HTTP і підтримує його як транспорт передачі даних від клієнта серверу і назад.

Сервлети в промисловому програмуванні використовуються для:

- прийому вхідних даних від клієнта;
- взаємодії з бізнес-логікою системи;
- динамамічній генерації відповіді клієнту.

2.8 AJAX

Згідно до минулого розділу веб-сервіс працює зі сторінками html. Недоліком стандартного підходу є те, що html-сторінка яку формує сервер є статичною, що зовсім не підходе для відстеження динамічних даних у реальному часі. Як рішення даної проблеми прийнято використовувати технологію AJAX.

AJAX (Asynchronous JavaScript And XML) — підхід до побудови користувацьких інтерфейсів веб-додатків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані.

Суть підходу

AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX-підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

- використання DHTML для динамічної зміни змісту сторінки;
- використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю;
- альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON);
- динамічне створення дочірніх фреймів.

Порівняння класичного підходу та AJAX

Класична модель веб-застосунку:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер надсилає запит серверу;
- у відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т. д.

Модель AJAX:

- цористувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер відправляє відповідний запит на сервер;
- сервер віддає тільки ту частину документу, яка змінилася.

Такий підхід повністю задовольняє потребам системи.

2.9 Проблема багатопоточності веб-сервісу

Згідно до розділу 2.7 (Принцип роботи кліент + сервер (Java Servlets)) для кожного нового користувача Java формує свій потік виконання. З одного боку – це надає деякі переваги. Наприклад, кожен для одного й того ж пристрою зможе обрати свої параметри для відстеження і спостерігати їх зміну. Значним недоліком цього підходу є сам цільовий пристрій, що не зможе обробити два чи більше запити, що прийдуть одночасно. Це приведе до спотворення даних і, як результат - виведення помилкової інформації.

Вирішення проблеми багатопоточності

Вирішити проблему можливо двома шляхами:

1. почерговий доступ до роботи з пристроєм;
2. розмежування сервісу на ролі.

Почерговий доступ до роботи з пристроєм

Мова програмування Java передбачає для таких цілей використання синхронізації.

Синхронізація – це процес, який дозволяє виконувати всі паралельні потоки в програмі синхронно. Синхронізація дозволяє уникнути помилок узгодженості пам'яті, викликаних непослідовним доступом до спільної пам'яті.

Коли метод (функція) оголошений як синхронізований - потік тримає монітор для об'єкта, метод якого виконується. Якщо інший потік виконує синхронізований метод, інший потік, що хоче отримати доступ заблокується до тих пір, поки інший потік не відпустить монітор.

Синхронізація досягається в Java використанням зарезервованого слова synchronized.

Даний спосіб має два недоліки:

1. синхронізовані методи в Java вносять додаткові витрати на продуктивність;
2. затримки для доступу до опитування.

Ці два недоліки пов’язані між собою і виникають із-за того, що віртуальна машина не дає доступ до секції коду оголошеного як synchronized іншим потокам, доки виконується потік, що знаходиться в секції, тим самим зупиняючи потоки, що очікують своєї черги. Це негативно впливає на роботу програми, оскільки JVM витрачає багато ресурсів на блокування. Також виходячи з того, що кожен потік має очікувати своєї черги, дані, що будуть отримані з проміжним сервером з великою ймовірністю будуть помилковими. Не дивлячись на частоту опитування, якщо клієнтів буде більше двох, ймовірність отримати помилкові дані - максимальна.

Виходячи з описаного вище, даний спосіб вирішення проблеми багатопоточного доступу до програми не є раціональним і не підходить для реалізації.

Розмежування сервісу на ролі

За допомогою додання у систему розмежування на ролі можливо досягти отримання коректних даних, тим що доступ до даних перетворювача частоти або будь якого іншого пристрою, що підтримує протокол Modbus TCP, зможе мати тільки адміністратор, а всі інші, хто захоче підключитись до пристрою, будуть переходити на сторінку з параметрами, які обрав адміністратор системи, оминаючи сторінку з входом до системи та вибором параметрів для відстеження. Досягається це тим, що тільки адміністратор відправляє запроси, після чого зберігає відповіді у контексті застосунка, а потоки всіх інших користувачів лише забирають відповіді з контексту, а не відправлять власні запити до цільового пристрою.

Дані адміністратора пропонується зберігати у файлах налаштувань або у реляційній базі даних (MySql, MardiaDb, MSSQL, тощо).

Недоліком даного способу є чітка прив’язка до параметрів обраних адміністратором системи, але це вигідно впливає на продуктивність системи, та гарантує отримання коректних даних.

3. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЦИФРОВИХ ЕЛЕКТРОПРИВОДІВ

3.1 Цільовий пристрій

Згідно до попередніх розділів система працює в мережі Modbus TCP та може підключатися до будь-якого пристрою, що підтримує даний протокол. Для тестування роботи системи був обраний перетворювач частоти компанії Schneider Electric Altivar Process. Даний вибір обумовлюється наступними причинами:

1. поширеність на виробництвах;
2. необхідність контролю;
3. велика кількість параметрів для відстеження;
4. наявність документації та специфікації.

3.2 Перетворювач частоти Altivar Process

Altivar Process - нова серія перетворювачів частоти компанії Schneider Electric, що дозволяє забезпечити якісне управління двигунами практично у всіх галузях промисловості. Забезпечує максимальну ефективність технологічного обладнання, володіє чудовими прикладними алгоритмами управління і комунікаційними можливостями. Залежно від вимог, пропонуються перетворювачі частоти настінного або підлогового виконання, а також комплектні системи управління зі ступенями захисту IP 21,IP 23, IP 54 і IP 55

Перетворювач Altivar Process був спеціально розроблений для промислових процесів, таких як:

- Водопідготовка та водовідведення: насосні станції, станції підвищення тиску, насоси свердловин, підйомні станції, станції аерації, для компресії і транспортування дренажу ...
- Нафтогазова промисловість: глибинні насоси, домкратні насоси, насоси для нафтопроводів, насоси підкачки, насоси для нагнітання води в пласт, компресори регазифікації ....
- Нафтопереробка: насоси, вентилятори, компресори ...
- Видобуток корисних копалин і металургійна промисловість: конвеєри, транспортери, вантажопідйомні механізми, дробарки ...
- Харчова промисловість: конвеєри, міксери, центрифуги, сушарки ...

Серія перетворювачів частоти Altivar Process розроблена для отримання максимальної продуктивності обладнання в поєднанні з винятковими алгоритмами управління двигунами і можливостями інтеграції в системи управління і контролю.

Пропонуються спеціалізовані прикладні функції для устаткування, яке застосовується в різних галузях промисловості:

- Можливість управління електродвигунами будь-яких типів.
- Робота в режимі "ведучий - ведений" при будь-якому способі з'єднання вузлів механізмів (жорстке або еластичне з'єднання).
- Мережеві сервіси забезпечують продовження роботи навіть у разі розриву з'єднання.
- Інтегрований веб-сервер і журнал подій зменшують час простою, дозволяючи швидко діагностувати і усунути несправність або виконати планове технічне обслуговування.

Залежно від виконання, перетворювачі частоти Altivar Process пропонуються в декількох варіантах виконання за типом монтажу та з різними ступенями захисту:

- Перетворювачі частоти для настінного монтажу, ступінь захисту IP 21, діапазон потужності від 0.75 кВт до 160 кВт, з можливістю монтажу без оболонки в електроприміщеннях, або з установкою в шафу.

- Перетворювачі частоти для настінного монтажу, ступінь захисту IP 55, діапазон потужності від 0.75 кВт до 90 кВт, готові до експлуатації в несприятливих умовах в приміщенні або при зовнішній установці для зменшення довжини кабелю двигуна. Перетворювачі частоти для настінного монтажу зі ступенем захисту IP 55 можуть комплектуватися роз'єднувачем.

- Перетворювачі частоти для підлогової установки, зі ступенем захисту IP 21 і IP 54, діапазон потужності від 110 кВт до 315 кВт, комплектний пристрій з мінімальними габаритними розмірами, з можливістю застосування в звичайних або несприятливих умовах навколишнього середовища

Для досліду був обраний перетворювач частоти для настінного монтажу зі ступенем захисту IP21 через його наявність у лабораторії НТУ Дніпровська Політехніка. Характеристики перетворювача наведені у таблиці 3.1.

Таблиця 3.1 – характеристики перетворювача частоти ATV 900

спосіб установки	виконання для настінного монтажу
Діапазон потужності, мережа 50/60 Гц (кВт)	
Трифазна, 200 – 240 В.	0.75 – 45
Трифазна, 200 – 240 В.	–
Трифазна, 200 – 240 В.	0.75 – 90
Електропривід	
Вихідна частота	0.1 – 500 Гц
Закон керування	
Асинхронний двигун	По п'яти точках, енергозберігаючий з контуром швидкості, в розімкнутої системі, в замкнутій системі
Синхронний двигун	СДПМ, високодинамічний СДПМ, СДПМ в замкнутій системі, реактивний двигун
Функції	• Ефективне управління двигуном з перевантаженням по моменту до 180% Мн в розімкнутої або замкнутої системі регулювання • Можливість роботи з асинхронними, синхронними і спеціальними двигунами, незалежно від класу ефективності і виробника. Двигуни з постійними магнітами, високомоментні двигуни, двигуни з конічним ротором, реактивні індукторні двигуни • Інтегрований здвоєний порт EtherNet / IP і Modbus TCP, кібербезпека (Achilles Level 2) • Зручність використання в системах автоматизації PlantStruxure і Foxboro Evo • Енергоефективність, контроль відхилення енергоспоживання системи від номінального значення • Адаптація до особливостей технологічного процесу • Вбудовані функції безпеки STO, сертифіковані по SIL3

	• Режим роботи «ведучий - ведений» і розподіл навантаження між перетворювачами частоти: 1. розподіл моментів при жорсткому з'єднанні елементів приводу 2. розподіл моментів при гнучкому поєднанні елементів приводу • Доступ до технічної документації за допомогою динамічного QR коду • Вимірювання в режимі реального часу з архівацією і можливістю настройки інформаційної панелі • Функції попередження про необхідність технічного обслуговування (наприклад, контролю температури • за допомогою датчиків PT100 / 1000, відстеження часу роботи вентиляторів)
Кількість аналогових входів	3: конфігуруються за напругою (0 - 10 В) або по струму (0 - 20 мА / 4 - 20 мА), включаючи два входи для підключення датчиків температури (PTC, PT100, PT1000 або KTY84)
Кількість дискретних входів	8: Напруга 24 В.
Кількість дискретних виходів	1: Програмований
Кількість аналогових виходів	2: конфігуруються за напругою (0 - 10 В) або по струму (0 - 20 мА)
Кількість релейних виходів	3: один з перекидним контактом і два з нормально відкритими контактами
Входи функцій безпеки	2: для функції безпеки STO
Модуль релейних входів	3: нормально відкриті контакти
Комунікаційні можливості	Здвоєний порт EtherNet / IP і Modbus / TCP, послідовний інтерфейс Modbus

3.3 Параметризація перетворювача частоти

Для налаштування пристрою для роботи з системою відстеження, потрібно перейти до підменю конфігурації вбудованого Ethernet (Рис. 3.1)

[Embd Eth Config] *E Ł E* **- Menu**

Access

[Communication] → [Comm parameters] → [Embd Eth Config]

Рисунок 3.1 Конфігурація вбудованого Ethernet

Необхідно налаштувати наступні параметри:

- IP Mode Ether. Embd: встановлює як задається IP-адреса ПЧ;
- IP address – значення адреси;
- Mask – маска підмережі;
- Gateway – адреса шлюзу.

Послідовність дій для налаштування ілюструється на рисунках 3.2-3.5

1. Перейдіть у головне меню

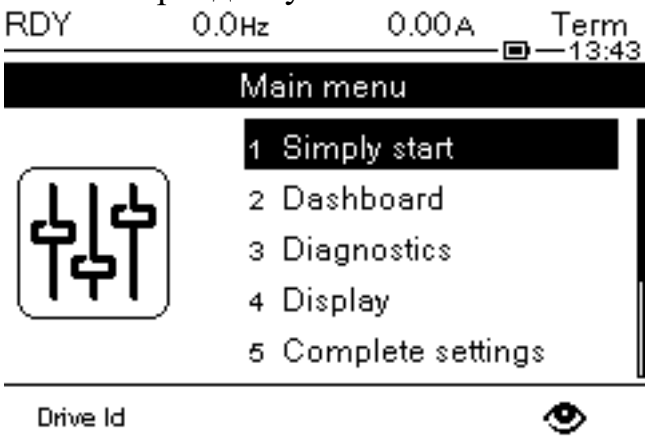

Рисунок 3.2 – Головне меню

2. Перейдіть до меню «комунікація»

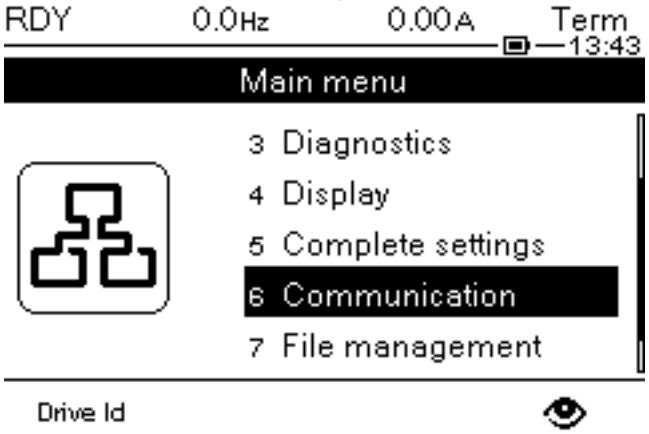

Рисунок 3.3 – Головне меню

3. Виберіть підменю «конфігурація вбудованого Ethernet»

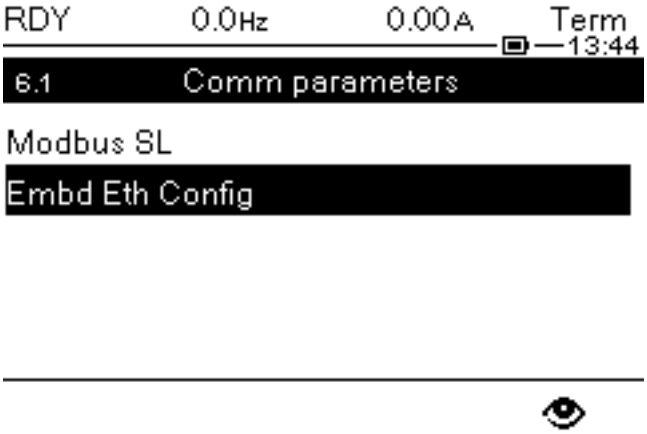

Рисунок 3.4 – Підменю «коммунікаційні параметри»

4. Введіть значення параметрів в даному підменю

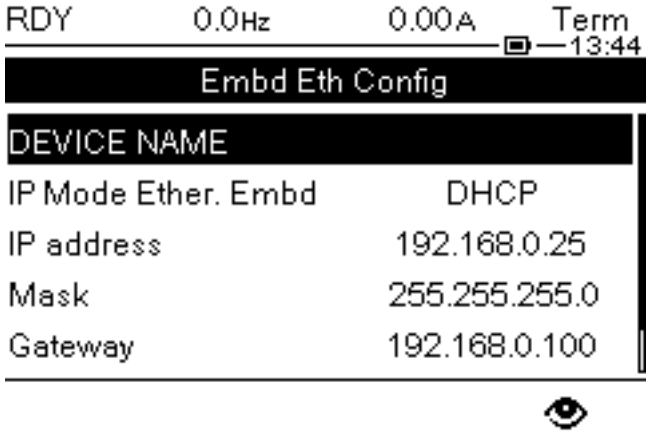

Рисунок 3.5 – Підменю «конфігурація вбудованого Ethernet»

3.4 Алгоритм роботи програмного забезпечення

На рисунку 3.5 наведена блок-схема роботи системи. Далі в цьому розділі, згідно з блок-схемою, буде стисло описаний кожен крок.

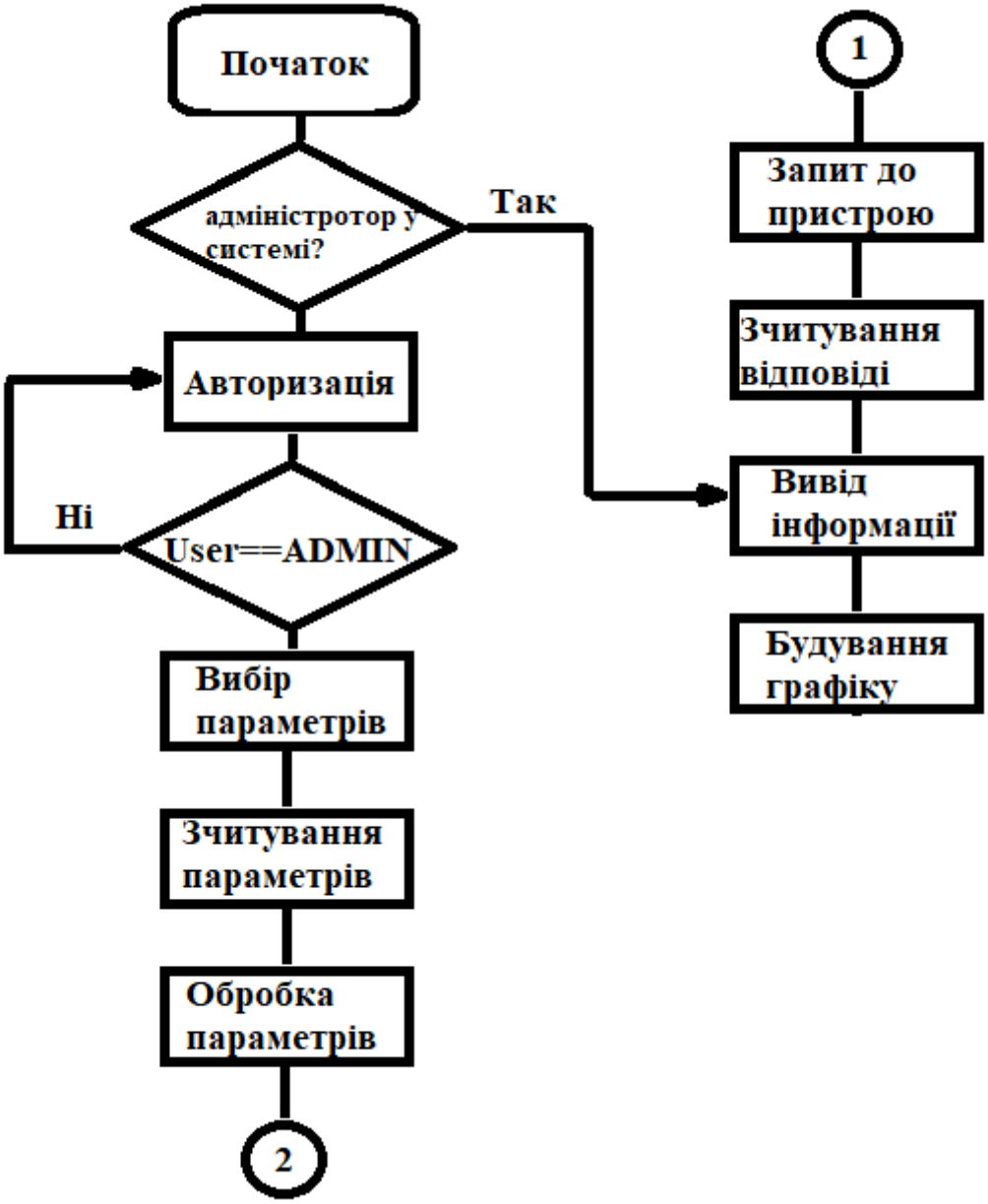

Рисунок 3.6 Блок-схема роботи системи відстеження.

3.5 Структура проекту

Проект має звичайну структуру веб додатку та містить у своєму складі наступні компоненти:

1. вихідний код;
2. ресурси;
3. HTML-сторінки для відображення та налаштування.

Ієрархічна структура приведена на рисунках 3.6-3.8

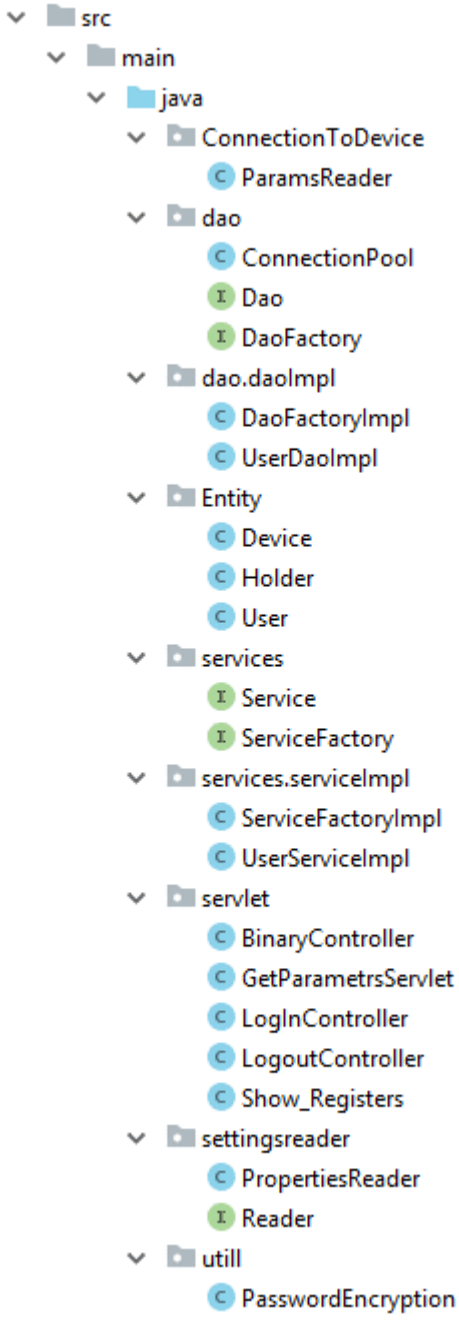

Рисунок 3.7 Структура пакетів з класами, що містять вихідний код

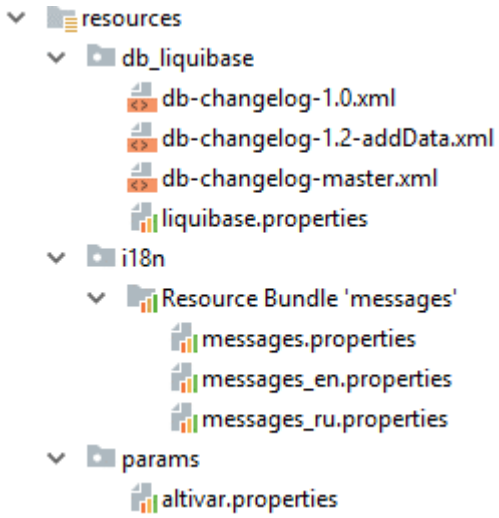

Рисунок 3.8 Структура пакетів з файлами ресурсів

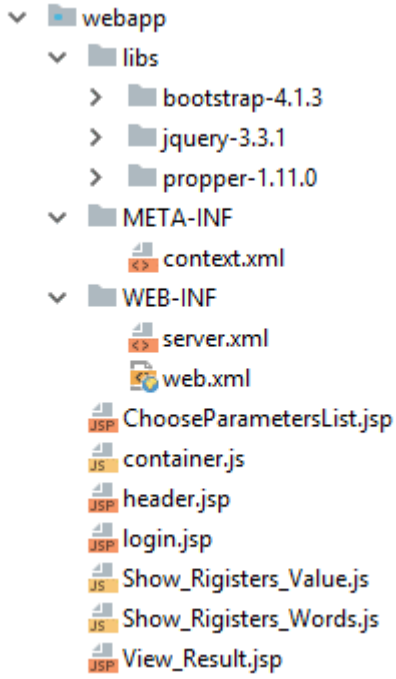

Рисунок 3.9 Структура пакетів з сторінками відображення та налаштуваннями

3.6 Запуск системи

Для початку функціонування системи, після налаштування пристрою, потрібно перейти до головної сторінки веб-серверу, що знаходиться за сконфігурованою адресою localhost: 8080.

3.7 Авторизація

Після переходу за вказаним посиланням потрібно пройти авторизацію у додатку. Вид додатку приведений на рисунку 3.9.

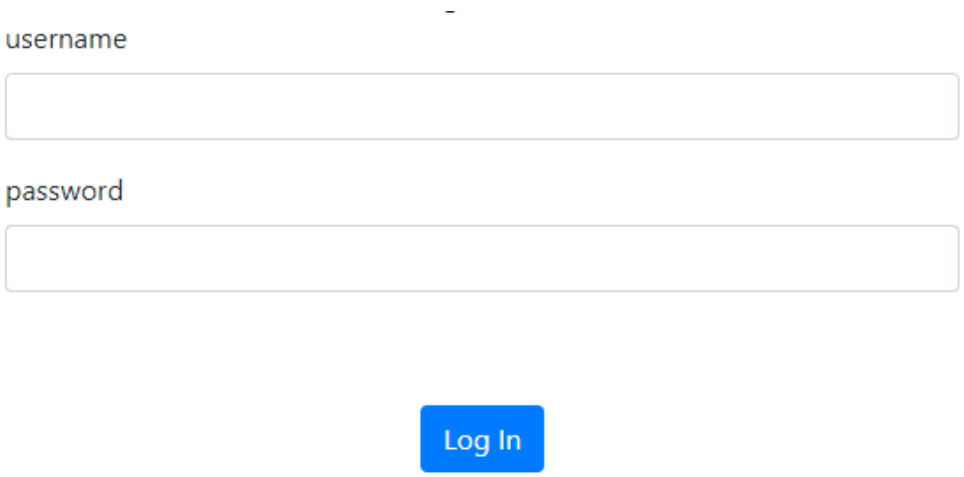

Рисунок 3.10 Форма авторизації

За обробку даних з форми відповідає клас контролер – LogInController. Сам контролер має три сценарії роботи:

1. Дані, що введені у форми валідні та відповідають реальному користувачу, що має права адміністратора.
2. Дані не валідні.
3. Адміністратор вже є у системі.

При першому сценарії перевіряється умова:

```
if (userFromDb != null &&
PasswordEncryption.checkPassword(password, userFromDb.getPassword())),
```

де:

userFromDb – модель реального користувача, що знаходиться у локальній базі даних.

PasswordEncryption – клас, необхідний для шифрування паролю та перевірки еквівалентності зашифрованого паролю до незашифрованого.

Якщо перевірка успішна – користувач заноситься до контексту додатку та зберігається у відкритій сесії. Якщо ж умова не виконується, спрацьовує другий сценарій.

При другому сценарії користувач перенаправляється на сторінку логіну, а до запиту додається повідомлення про некоректність даних, після чого сторінка логіну прийме вигляд зображений на рисунку 3.10.

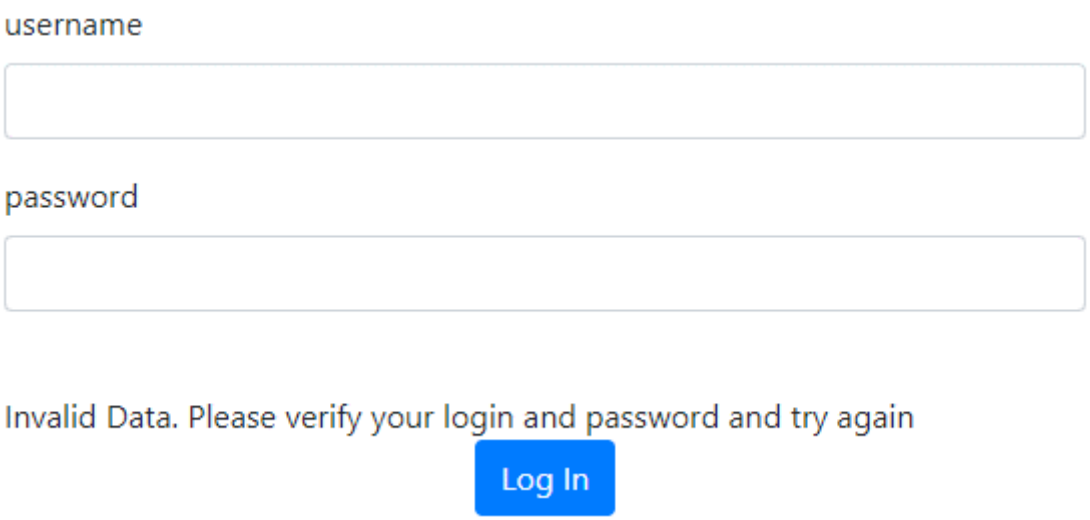

Рисунок 3.11 Форма повторної авторизації

Третій сценарій передбачає, що адміністратор вже є у системі. Для цього перевіряється наступна умова:

```
if (this.getServletContext().getAttribute("User") == null)
```

Якщо умова виконується і адміністратора немає у системі, то починається перевірка першого сценарію. Якщо ж умова не виконується, то користувач одразу перенаправляється до сторінки з відображенням даних, які обрав адміністратор.

Причини необхідності даного сценарію описані у розділі 2.9 «Проблема багатопоточності веб-сервісу».

3.8 Вибір даних

При успішному проходженні авторизації користувач перенаправляється до сторінки вибору параметрів для відстеження. Зовнішній вигляд сторінки ілюструється на рисунку 3.11.

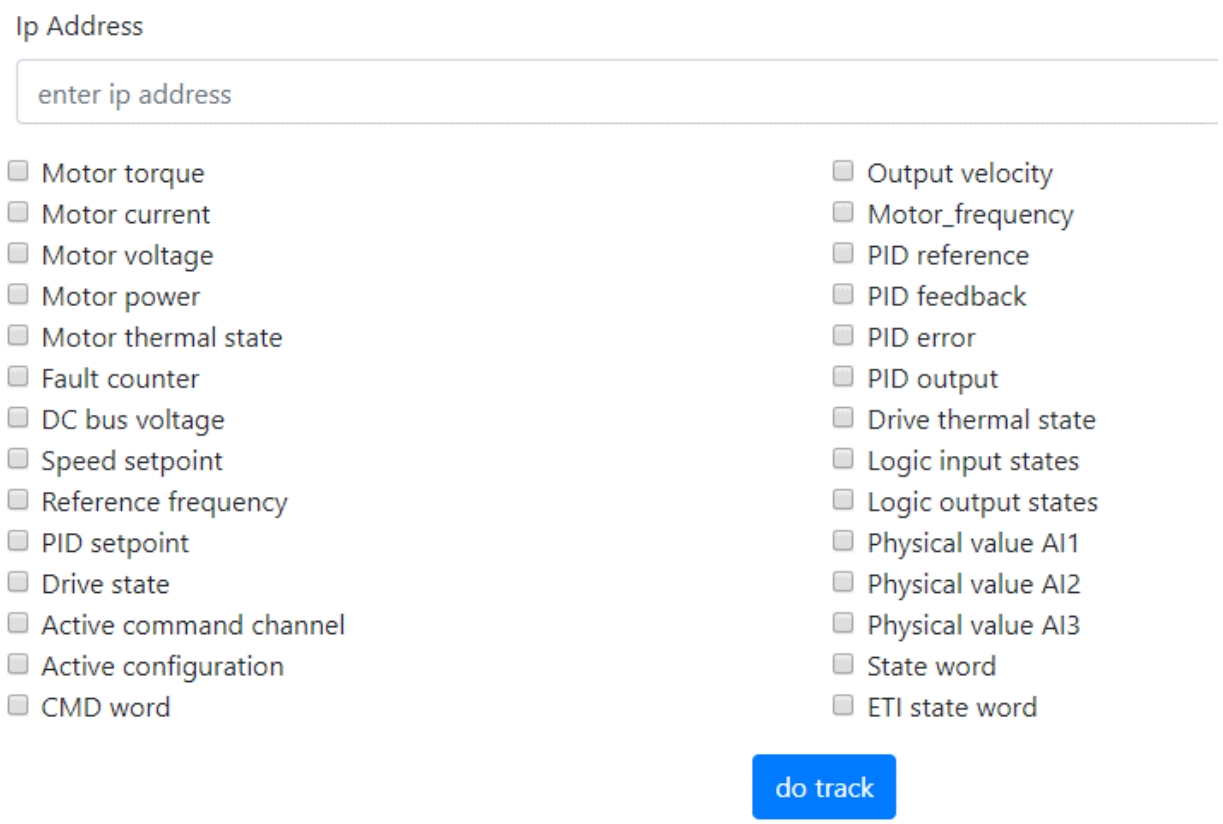

Рисунок 3.12 Зовнішній вигляд сторінки вибору параметрів

Для проходження до наступного етапу, потрібно ввести адресу пристрою згідно до розділу 3.3 «Параметризація перетворювача частоти», та відмітити необхідні параметри для відстеження.

За прийняття даних, що надходять з даної сторінки, відповідає клас контролер GetParametersServlet. Він виконує наступні дії:

1. Зчитує адресу, що надійшла зі сторінки

```
String ip = (String) request.getAttribute("ip");
```

2. Зчитує список параметрів, що обрав користувач

```
List<String> list = Arrays.asList(request.getParameterValues("gridRadios"));
```

3. Визиває функцію, що основуючись на списку формує хеш-таблицю для якої ключем є - назва параметру, а значенням - адреса регістру.

```
HashMap map = ChooseRegisters(list);
```

Для подальшого розширення додатку та уникнення прив’язки до конкретного пристрою всі параметри зберігаються у окремих файлах, що знаходяться у ресурсах додатку та мають розширення .properties. Так для налаштування системи для роботи з іншим пристроєм потрібно додати новий

файл властивостей. В якому параметри повинні знаходитись за наступним шаблоном:

```
Motor_torque = val:3205
Drive_state = word:3240
```

де:

Частина перед знаком рівності – назва параметру

Частина після знаку рівності – тип та адреса регістру

Після проходження всіх дій хеш-таблиця зберігається у сесії, а користувач перенаправляється до сторінки відображення параметрів.

3.9 Опитування приладу

В момент переходу на сторінку відображення даних одночасно спрацьовують два скрипти, що написані на мові javaScript. Головним завданням скриптів є – за допомогою ajax збуджувати контролери Show_Registers та BinaryController та будувати таблиці з даних, що прийшли від контролерів.

Show_Registers

Головним завданням контролеру є опитування пристрою, збереження даних у форматі JSON (Java Script Object Notation) та відповідь до скрипту, що його викликає. За опитування пристрою відповідає об’єкт класу ParamsReader, що оснований на некомерційній java-бібліотеці wimpi.

За допомогою внутрішнього об’єкту ParamsReader встановлюється з’єднання з цільовим приладом. У зв’язку з великою частотою опиту пристрою доцільно не відкривати нове з’єднання при кожному підключенні до пристрою, а тримати його відкритим протягом усього сеансу роботи з додатком.

Відповідно до приведеної інформації, за роботу з пристроєм відповідає внутрішній об’єкт контролеру. Сигнатура функції для опитування цільового пристрою має наступний вигляд:

```
public Map<String, List<Device>> CheckValues(HashMap map, TCPMasterConnection con),
```

де:

`public` – модифікатор доступу, що дає можливість створити об’єкт даного класу у будь-якій частині додатку;

`Map<String, List<Device>>` - параметризований результат функції, що являє собою таблицю – ключ , значення. Ключем для таблиці будуть виступати строки – values та words, а значенням для них списки об'єктів класу «Device».

Даний клас призначений для зберігання та утримання параметрів, що надходять з пристрою по протоколу Modbus TCP.

`HashMap map` - об'єкт сформований на минулих етапах, що являє собою хеш - таблицю, ключем якої є – назва параметру, а значенням – адреса регістру.

`TCPMasterConnection con` - Об'єкт з'єднання з пристроєм.

Віповідно до минулого розділу, з'єднання з пристроєм відкривається один раз і існує протягом всього сеансу користування додатком. Для відкриття з'єднання використовується функція, що має наступну сигнатуру:

```
public void ConnctionTo(String addres),
```

де:

`void` - зарезервоване слово мови програмування java, яке вказує на те, що функція не повертає результат;

`String addres` – ip-адреса пристрою, вказана користувачем;

Для підключення функція використовує наступні команди:

```
InetAddress addr = InetAddress.getByName(addres);
con = new TCPMasterConnection(addr);
con.setPort(502);
con.setTimeout(1000);
con.connect();,
```

де:

`InetAddress.getByName(addres)` - реєстрація id-адреси пристрою;

`con` `= new TCPMasterConnection(addr)` – створення об'єкту класу TCPMasterConnection;

`con``.setPort(502)` – встановлення порту на ПК для з'єднання з пристроєм (за замовченням зарезервований під Modbus TCP) ;

`con``.setTimeout(1000)` - встановлення максимального часу очікування відповіді від пристрою;

`con``.connect()` – з'єднання з пристроєм.

Для отримання об'єкту з'єднання використовується функція з наступною сигнатурою:

```
public TCPMasterConnection GetConnection()
```

Функція перевіряє чи існує з'єднання з пристроєм і, при успішній перевірці, повертає об'єкт з'єднання.

Опитування даних основується на ітерації по хеш-таблиці та запиту даних з пристрою по ключу з цієї таблиці. Для прямого опитування пристрою використовується функція з наступною сигнатурою:

```
private ReadMultipleRegistersResponse getResponse(TCPMasterConnection
con, String key),
```

де:

private - модифікатор доступу, який дозволяє використовувати дану функцію тільки у внутрішніх цілях класу;

`ReadMultipleRegistersResponse` – об'єкт, що буде сформований з відповіді перетворювача частоти або будь-якого іншого пристрою, що підтримує Modbus TCP.

Для опитування пристрою використовуються наступні команди:

```
ReadMultipleRegistersRequest request =
new ReadMultipleRegistersRequest(key, 1);

ModbusTCPTransaction trans = new ModbusTCPTransaction(con);
trans.setRequest(request);
trans.setRetries(10);
return trans.getResponse();,
```

де:

new `ReadMultipleRegistersRequest(key, 1)` – створення нового об'єкту запиту до пристрою, що має зчитати один регістр адреса якого знаходиться в змінній – key;

new `ModbusTCPTransaction(con)` - створення нової транзакції у відкритому підключенні;

`trans.setRequest(request)` - встановлення запиту;

`trans.setRetries(10)` - встановлення максимальної кількості повторних запитів, при відсутності відповіді від пристрою;

return `trans.getResponse()` - повернення відповіді від пристрою.

Після відповіді від пристрою перевіряється до якого типу параметрів належить значення регістру.

Якщо параметр – звичайне цифрове значення, у список «values» заноситься новий об'єкт Device, ім'ям якого буде назва параметру, а значенням – відповідь з пристрою:

```
values.add(new Device(String.valueOf(entry.getValue()),
String.valueOf(response.getRegisterValue(0))));
```

У разі, коли регістр відноситься до типу WORD – у список «words» заноситься новий об'єкт Device, ім'ям якого буде назва параметру, а значенням – відповідь з пристрою, що конвертована у двійковий формат:

```
words.add(new Device(String.valueOf(entry.getValue()),
Integer.toString(response.getRegisterValue(0), 2)));
```

Після проходження таким чином всієї хеш – таблиці, створюється нова таблиця, що являє собою два списки об'єктів «Device» та ключи, за якими їх можливо дістати – «words» та «values».

Дані, що отримав контролер потрібно відправити до html сторінки. Для передачі даних між двома мовами програмування був обраний стандарт JSON. Щоб перейти від даних на мові Java до даних, що розуміє JavaScript, потрібно скористатися бібліотекою Gson, що поширюється компанією Google. Перетворення відбувається наступним чином:

```
Map<String, List<Device>> allParams = t_read.CheckValues(map, con);
String JSON = builder.toJson(allParams.get("values"));
```

Також контролер підтримує інший сценарій, при якому адміністратор вже обрав необхідні параметри. Для того щоб не ініціювати повторне опитування перетворювача частоти, потрібно місце, в якому можливо зберегти проміжні дані, що сформовані за запитом адміністратора, а всім бажаючим – видавати значення, що зберігаються у тому місці. Для реалізації був створений статичний (існує в єдиному екземплярі) клас Holder. Так головною задачею даного класу є – зберігання списків «values» та «words»:

```
Holder.setValues(allParams.get("values"));
Holder.setWords(allParams.get("words"));
```

Після збереження та конвертування всіх необхідних даних контролер формує відповідь до скрипту, що його визиває.

BinaryController

Даний контролер приводиться в дію за допомогою скрипта, що написаний на мові JavaScript та має назву – Show_Registers_Words.js. Даний контролер не збуджує опитування пристрою. Контролер повністю опирається на дані, що були отримані при опитуванні контролером «Show_Registers» і лише зчитує інформацію, що знаходиться в класі Holder:

```
List words = Holder.getWords();
```

Далі перевіряється умова, що об’єкт існує та не порожній. В разі виконання умови, дані конвертуються до формату JSON і контролер формує відповідь до скрипту, який його викликав.

3.10 Відображення даних

Відповідно до розділу 3.9, у додатку є два скрипта, що спонукають до дії контролери. Так, одним із головних завдань скрипта, є будування динамічної таблиці, згідно з даними, що прийшли як відповідь з контролера. За будування таблиці з звичайними цифровими даними відповідає скрипт Show_Rigisters_Value.js, а за будування таблиці з даними представленими у двійковому форматі – Show_Rigisters_Words.js.

Сторінка виводу інформації ілюструється на рисунку 3.12.

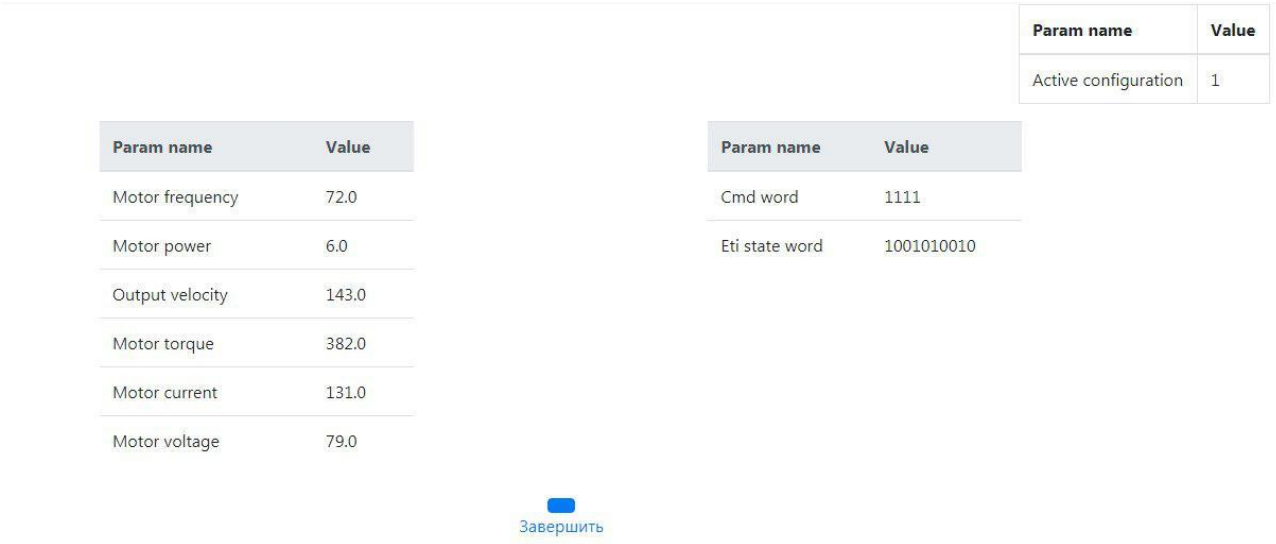

Рисунок 3.13 – Сторінка виводу інформації

3.11 Графік

Для відображення даних у реальному часі у системі передбачено побудова графіку. Графік будується за залежністю $y = f(t)$ і оновлюється з частотою 10 Гц. Приклад графіку зображено на рисунку 3.13.

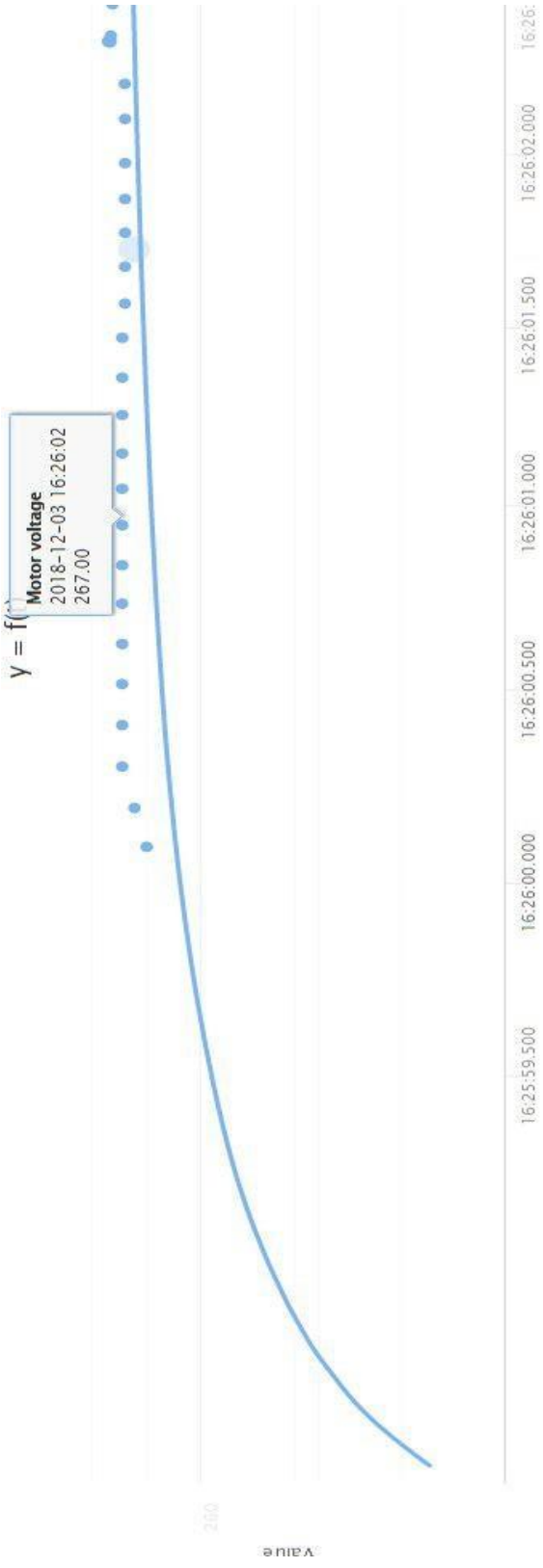

Рисунок 3.14 Приклад часової діаграми напруги

3.12 Збереження даних

Після завершення роботи відбудеться перехід до сторінки збереження даних. На даній сторінці пропонується зберегти дані до файлу (Рис. 3.14).

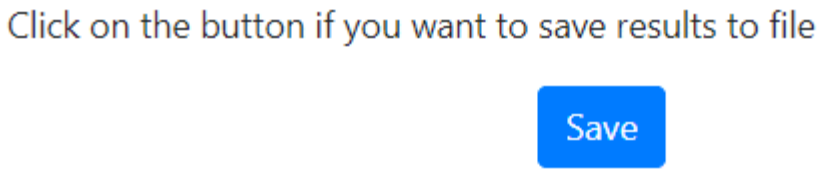

Рисунок 3.15 Сторінка збереження даних

Після натискання на кнопку відбудеться збереження останніх 300 значень, що були зчитані з перетворювача частоти. Зовнішній вигляд файлу зображено на рисунку 3.15.

```
Motor torque
3106 Sat Dec 01 15:23:41 EET 2018
3106 Sat Dec 01 15:23:41 EET 2018
3107 Sat Dec 01 15:23:41 EET 2018
3107 Sat Dec 01 15:23:41 EET 2018
3107 Sat Dec 01 15:23:41 EET 2018

Motor voltage
3398 Sat Dec 01 15:23:41 EET 2018
3398 Sat Dec 01 15:23:41 EET 2018
3398 Sat Dec 01 15:23:41 EET 2018
3399 Sat Dec 01 15:23:41 EET 2018
3399 Sat Dec 01 15:23:41 EET 2018

Motor current
3009 Sat Dec 01 15:23:41 EET 2018
3009 Sat Dec 01 15:23:41 EET 2018
3010 Sat Dec 01 15:23:41 EET 2018
3012 Sat Dec 01 15:23:42 EET 2018
```

Рисунок 3.16 Зовнішній вигляд файлу з даними.

4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Ринок праці вимагає підготовки кваліфікованих магістрів спеціальності 141 «Електромеханічні системи та електропривод». Навчання за даним напрямком включає в себе вивчення електродвигунів та систем керування, відстеження та автоматизації. Дані системи мають назву – SCADA та відіграють велику роль на сучасних виробництвах, що свідчить про необхідність досконалого вивчення даних систем.

SCADA (supervisory control and data acquisition, диспетчерське управління і збір даних) - програмний пакет, призначений для розробки або забезпечення роботи в реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або управління. SCADA може бути частиною АСУ ТП, АСКОЕ, системи екологічного моніторингу, наукового експерименту, автоматизації будівлі і т. Д. SCADA-системи використовуються у всіх галузях господарства, де потрібно забезпечувати операторський контроль за технологічними процесами в реальному часі.

Підсистеми, що входять до складу SCADA-системи:

1. драйвери або сервери введення-виведення - програми забезпечують зв'язок SCADA з промисловими контролерами;
2. система реального часу - програма, яка забезпечує обробку даних в межах заданого часу з урахуванням пріоритетів;
3. людино-машинний інтерфейс - інструмент, який представляє дані про хід процесу людині оператору, що дозволяє оператору контролювати процес і керувати ним.

Основні функції:

1. ведення бази даних реального часу;
2. виконання розрахунків;
3. графічне представлення даних і параметрів у вигляді мнемосхем, графіків, діаграм і т.д;
4. попереджувальна сигналізація;
5. архівування інформації;
6. генерування звітів.

Метою проекту є створення системи відстеження, що може бути аналогом системам, що знаходяться у середньому ціновому діапазоні та може використовуватися в учбових цілях у наступних дисциплінах:

1. системи керування електроприводами;
2. автоматизація загальнопромислових установок і технологічних комплексів;
3. комп'ютерні технології в задачах електромеханіки.

Власноручне виготовлення даної системи надає ряд переваг, серед яких:

1. незалежність від компанії, що розроблює пристрій;
2. незалежність від платформи;
3. створення програми на сучасній інформаційній базі;
4. можливість розширення та вдосконалення.

В ході проекту була розроблена документація та програмне забезпечення для відстеження стану цільового пристрою.

В розділі «економічне обґрунтування» необхідно визначити:

1. Працеємкість опрацювання ПЗ.
2. Розрахунок витрат на створення програмного виробу.

4.1 Визначення працеємкості опрацювання програмного продукту

Нормування праці в процесі створення ПЗ істотно ускладнено через творчій характер праці програмістів. Тому працеємкість опрацювання ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Працеємкість опрацювання ПЗ можна розрахувати за формулою:

$$t = t_0 + t_и + t_а + t_п + t_{опл} + t_д, \text{ людино-годин,} \quad (1)$$

де t_0 – витрати праці на підготовку і опис поставленого завдання;

$t_и$ – витрати праці на дослідження алгоритму вирішення завдання;

$t_а$ – витрати праці на опрацювання блок-схеми алгоритму;

$t_п$ – витрати праці на програмування за готовою блок-схемою;

$t_{опл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації по завданню.

Складові частини витрат праці визначаються на підставі умовної кількості операторів в ПЗ, що опрацьовується, в число яких входять ті оператори, які необхідно написати в процесі роботи над програмою з урахуванням можливих уточнень в постановці завдання і удосконалення алгоритму.

Умовна кількісті операторів в програмі:

$$Q = q * c * (1 + p), \qquad (2)$$

де q – кількість операторів, що припускається;

c – коефіцієнт складності програми;

p – коефіцієнт корекції програми в процесі її опрацювання.

Згідно до формули (2) умовна кількість операторів у програмі становить:

$$Q = 100 * 1.5 * (1 + 0.07) = 160;$$

Витрати праці на вивчення опису завдання визначаються з урахуванням уточнення опису і кваліфікації програміста за формулою:

$$t_{\text{и}} = \frac{Q*B}{(75...85)*k}, \text{ людино-годин}, \quad (3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання, В = 1,2…1.5;

k – коефіцієнт кваліфікації програміста, що визначається залежно від стажу роботи за фахом.

Згідно до формулі (3) витрати праці на вивчення опису завдання та дослідження алгоритму становлять:

$$t_{\text{и}} = \frac{160*1,2}{75*0,8} = 3,2, \text{ людино-годин}$$

Витрати праці на опрацювання алгоритму рішення завдання:

$$t_{\text{а}} = \frac{Q}{(20...25)*k}, \text{ людино-годин} \quad (4)$$

Згідно до формули (4) витрати праці на опрацювання алгоритму рішення завдання становлять:

$$t_{\text{а}} = \frac{16}{22*0{,}8} = 9, \text{ людино-годин}$$

Витрати праці на складання програми за готовою блок-схемою:

$$t_{\text{п}} = \frac{Q}{(20...25)*k} = \frac{16}{22*0{,}8} = 9 \text{ , людино-годин} \qquad (5)$$

Витрати праці на налагодження програми на ЕОМ розраховуються за формулами:

$$t_{\text{отл}} = \frac{Q}{(4...5)*k} = \frac{16}{4*0{,}8} = 50, \text{ людино-годин;} \qquad (6)$$

Витрати праці на підготовку документації по завданню визначаються за формулою:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{ людино-годин,} \quad (7)$$

де $t_{\text{др}}$- працеємкість підготовки матеріалів до рукопису;

$$t_{\text{др}} = \frac{Q}{(15...20)*k} = \frac{16}{16*0{,}8} = 12{,}5, \text{ людино-годин}$$

$t_{\text{до}}$- працеємкість редагування, друку та оформлення документації;

$$t_{\text{до}} = 0{,}75 * t_{\text{др}} = 0{,}75 * 1{,}25 = 9{,}37, \text{ людино-годин}$$

Згідно до формули (7) витрати на підготовку документації по завданню становлять:

$$t_{\text{д}} = 12{,}5 + 9{,}37 = 21{,}81, \text{ людино-годин}$$

Згідно до формули (1) працеємкість опрацювання програмного забезпечення становитиме:

$$t = 1 + 3{,}2 + 9 + 9 + 50 + 21{,}8 = 94, \text{ людино-годин}$$

4.2 Розрахунок витрат на створення програмного виробу

Витрати на створення програмного виробу $k_{\text{пз}}$ включають витрати на заробітну плату виконавця програми $З_{\text{зп}}$ і вартість машинного часу, необхідного для налагодження програми на ЕОМ $З_{\text{мв}}$:

$$k_{\text{пз}} = З_{\text{зп}} + З_{\text{мв}} \text{ , грн.,} \qquad (8)$$

Заробітна плата виконавця визначається за формулою:

$$З_{зп} = t * С_{пр},\ \text{грн.},\quad (9)$$

де t - загальна працеємкість опрацювання ПЗ, людино-годин;

$С_{пр}$- середня годинна заробітна плата програміста.

Згідно до формули (9) заробітна плата виконавця становить:

$$З_{зп} = 94 * 100 = 9400, \text{грн}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * С_{мч}, \text{грн.},\quad (10)$$

де $t_{отл}$- працеємкість відладки програми на ЕОМ, людино-годин;

$С_{мч}$- вартість машино-години ЕОМ, грн./год.

Згідно до формули (10) вартість машинного часу, необхідного для налагодження програми на ЕОМ становить:

$$З_{мв} = 50 * 10 = 500, \text{грн}$$

Згідно до формули (8) витрати на створення програмного виробу становлять

$$k_{пз} = 9400 + 500 = 9900, \text{грн}$$

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП або іншої апаратури автоматика.

Очікувана тривалість опрацювання ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{міс.},\quad (11)$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу.

Згідно до формули (11) очікувана тривалість опрацювання ПЗ становить:

$$T = \frac{94}{1*176} = 0{,}53, \text{міс.},$$

Висновок

При розрахунку економічної складової магістерської роботи було встановлено, що витрати на створення програмного виробу становлять - 9900 гривень, а тривалість опрацювання програмного забезпечення становить – робочих 15 днів. Розрахунки свідчать про те, що одноразовий вклад у розробку ПЗ є цілком виправданим.

Висновок

В процесі розробки роботи були створенні наступні розділи:

1. «Завдання на розробку».
2. «Обґрунтування технічних рішень».
3. «Розробка системи моніторингу цифрових електроприводів».
4. «Техніко-економічне обґрунтування».

В розділі «Завдання на розробку» опрацьовані наступні пункти:

- висунуті головні та додаткові вимоги;
- побудована функціональна схема;
- приведені можливі контрольовані параметри.

В розділі «Обґрунтування архітектурних рішень» опрацьовані наступні пункти:

- обраний протокол передачі даних;
- обрана мова програмування;
- обраний спосіб виводу інформації;
- обраний спосіб вирішення проблеми багатопоточності.

В розділі «Розробка системи моніторингу цифрових електроприводів» опрацьовані наступні пункти:

- описаний цільовий пристрій;
- розроблено алгоритм роботи програмного забезпечення;
- приведені можливості та реалізація основних блоків коду;
- приведений спосіб збереження даних.

В розділі «Техніко-економічне обґрунтування» опрацьовані наступні пункти:

- визначена працеємкість опрацювання програмного продукту;
- розраховані витрати на створення програмного продукту.

Список літератури

1. Эккель Б. Философия Java. 4-е полное изд. – СПб.: Питер, 2015. – 1168 с.: ил. – (Серия «Классика computer science»).
2. ATV900 - Communication parameters addresses V1.9. – Режим доступу: https://www.schneider-electric.com/en/download/document/NHA80944/
3. Provides an object oriented Modbus implementation in Java. – Режим доступу: http:// jamod.sourceforge.net/api/net/wimpi/modbus/package-summary.html
4. Modbus Specifications and Implementation Guides. – Режим доступу: http: //www.modbus.org/specs.php
5. ATV900 Programming manual (V1.8). – Режим доступу: http://www.schneider-electric.ua/ru/download/document/NHA80757/
6. Ajax API documentation– Режим доступу: https://api.jquery.com/category/ajax/
7. Highcharts, Highstock and Highmaps documentation– Режим доступу: https://www.highcharts.com/docs
8. Методические указания к выполнению экономической части дипломной работы для студентов направления подготовки 6.050702 «Электромеханика» / Составители: Л.В. Тимошенко, И.В. Шереметьева – Днепропетровск: НГУ, 2015. – 15 с.