Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

(тема)
Моделювання автоматизованого комплексу безтарного сховища сировини сипучих матеріалів

Виконав: здобувач 2 року навчання, КІТПВм-23-3 гр.
Левченко К.О.
(прізвище, ініціали)

Спеціальність
174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

освітньої програми Комп'ютерно-інтегровані технологічні процеси і виробництва
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник доц.каф. КІТАМ Разумов-Фризюк Є.А.
(посада, прізвище, ініціали)

Допускається до захисту
зав. кафедри _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерно-інтегровані технологічні процеси і виробництва

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Левченку Кирилу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання автоматизованого комплексу безтарного сховища сировини сипучих матеріалів
затверджена наказом по університету від 22.11. 2024 р. № 1231 Ст
2.Термін подання здобувачем роботи до екзаменаційної комісії 10. 01. 2024 р.
3. Вихідні дані до роботи Комплекс безтарного сховища, з контролером Siemens, програмне забезпечення.

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз конструкції та складових комплексу
4.2 Аналіз основних цілей та задача комплексу
4.3 Розробка програмного забезпечення роботи комплексу
4.4 Розрахунок надійності комплексу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – с. формату А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

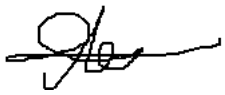
№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Загальний огляд комплексу	10.09.2024 – 21.09.2024	виконано
2	Аналіз основних елементів комплексу	01.10.2024 – 10.10.2024	виконано
3	Аналіз цілей та задач комплексу	15.10.2024 – 25.10.2024	виконано
4	Розробка алгоритму роботи комплексу	01.11.2024 – 10.11.2024	виконано
5	Розробка програмного забезпечення керування двигунами	20.11.2024 – 30.11.2024	виконано
6	Розробка програми автоматичної роботи комплексу	01.12.2024 – 10.12.2024	виконано
7	Розробка людино-машинного інтерфейсу	15.12.2024 – 20.12.2024	виконано
8	Розрахунок надійності комплексу	20.12.2024 – 28.12.2024	виконано

Дата видачі завдання 25 листопада 2024 р.

Студент _____ Левченко К.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц.каф. КІТАР Разумов-Фризюк Є.А.
(підпис) (посада, прізвище, ініціали)

Я , як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

10 січня 2025р. Левченко К.О. 

РЕФЕРАТ

Пояснювальна записка: 87 с., 3 табл., 86 рис., 22 джерел.

КОМПЛЕКС БЕЗТАРНОГО СХОВИЩА, АВТОМАТИЗАЦІЯ, КОНТРОЛЕРИ SIEMENS, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ

Об’єкт дослідження – Технології зберігання та транспортування сировини автоматизованими системами.

Предмет дослідження – Програмне забезпечення та людино-машинний інтерфейс для комплексу безтарного сховища сировини.

Мета роботи – розробка комплексу для зв’язку двох виробничих ліній та зменшення використання складських приміщень, що призведе до економічного покращення установи. Підібрати гнучку та варіативну компонентну базу, яка може бути змінена на вимогу замовника. Розробити адаптивне програмне забезпечення та зрозумілий людино-машинний інтерфейс.

У роботі розглянуто будову комплексу та його основні компоненти, що використовуються. Розглянуто основні цілі та задачі комплексу, економічні переваги.

Розроблено програму для керування кожним окремим вузлом комплексу та його елементом та розроблену програму для автоматичної роботи комплексу. Створено людино-машинний інтерфейс для взаємодії з комплексом операторами.

Розраховано надійність комплексу та вірогідність безвідмовної роботи комплексу.

Отримані результати та висновки по роботі можно віднести до 12 цілі сталого розвитку пункту 12.4 «Зменшити обсяг утворення відходів і збільшити обсяг їх переробки та повторного використання на основі інноваційних

технологій та виробництв», адже при подібному підході робота з сировиною стає зручнішою та якіснішою, що дозволяє більш якісно підійти до її зберігання та підготовки до використання, що в свою чергу дозволяє отримувати більш якісний вторинний продукт. Зберігання на малих складських площах дозволяє уникати утворені великих засмічених площ, що позитивно впливає на екологію.

ABSTRACT

Explanatory note: 87 pages, 3 tables, 86 figures, 22 sources.

BULK STORAGE COMPLEX, AUTOMATION, SIEMENS CONTROLLERS, SOFTWARE DEVELOPMENT FOR AUTOMATION

Object of research is technologies of storage and transportation of raw materials by automated systems

The subject of research is software and human-machine interface for a complex of bulk storage of raw materials.

The purpose of work is development of a complex to connect two production lines and reduce the use of warehouse space, which will lead to an economic improvement of the institution. Choose a flexible and variable component base that can be changed at the request of the customer. Develop adaptive software and a clear human-machine interface.

The structure of the complex and its main components used are considered in the work. The main goals and objectives of the complex, economic advantages are considered.

A program has been developed to control each individual node of the complex and its element, and a program has been developed for the automatic operation of the complex. A human-machine interface was created for interaction with the complex by operators.

The reliability of the complex and the probability of trouble-free operation of the complex are calculated.

The obtained results and conclusions of the work can be attributed to the 12 goals of sustainable development, point 12.4 "Reduce the volume of waste generation and increase the volume of its processing and reuse based on innovative technologies and productions", because with such an approach, working with raw materials becomes

more convenient and of higher quality, which allows for a more qualitative approach to its storage and preparation for use, which in turn allows to obtain a higher quality secondary product. Storage in small storage areas allows you to avoid the formation of large littered areas, which has a positive effect on the environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМЕНІВ

ПЗ – програмне забезпечення;

HMI – human-machine interface;

HART – highway Addressable Remote Transducer;

FBS – function block sequence;

STL – statement list;

ДСТУ – державний стандарт України;

PROFINET – протокол зв’язку Siemens;

MTBF – середній час напрацювання на відмову ;

MTBR – середінй час відновлення;

IEC 61508 – міжнародний стандарт оцінки надійності автоматичних систем управління;

SIL – safety integry level;

ПІД – пропорційно інтегрально диференційний регуялтор.

ВСТУП

Одною з величезних проблем будь-якого підприємства є нехватка складських площ та приміщень.

Ніяка установа не хоче тримати та тим самим займати площі матеріалами чи сировиною, так як вона само по собі не приносить економічної вигоди.

Отже виникає потреба в зберіганні того ж самого об’єму матеріалу але використовуючи менші площа.

Деякі установи також мають проблему зв’язку двох виробничих ліній між собою, коли одна лінія має виробляти сировину для іншою, отже виникає проблема проміжного сховища, без використання складських приміщень, яке може транспортувати сировину з однієї лінії в іншу на її вимогу.

Така система має бути не тільки зручною у використанні, тобто мати зручний інтерфейс взаємодії для операторів, а й мати досить високі показники варіативності та гнучкість компонентної бази, що означає що система має підлаштовуватися під вимоги кожного замовника або установи.

Програма для такої системи, у зв’язку з її гнучкістю, має мати можливість легко адаптуватися та змінюватися в залежності від поточних потреб та запитів.

Розробивши таку систему, ми отримаємо економічне вигідне та сучасне рішення для зберігання сировини на промислових підприємствах, можливість зручної роботи з сировиною та можливість інтеграції різних видів фільтрації та сепарації, а також екологічне рішення для зберігання брудних відходів чи вторинної сировини.

1 АНАЛІЗ БУДОВИ ТА КОМПОНЕНТНОЇ БАЗИ КОМПЛЕКСУ БЕЗТАРНОГО СХОВИЩА

1.1 Опис автоматичного комплексу безтарного сховища сировини

Автоматичний комплекс безатрного сховища речовини – це рішення для зберігання сипучих матеріалів (зерна, каміння, флекси) без використання значних складських приміщень та спеціалізованих ємностей таких як біг-беги. Окрім основної задачі, присутня можливість змішування та приготування міксів з таких матеріалів за допомогою спеціальної ємності (міксеру) з різних ємностей зберігання.

Універсальність такого комплексу забезпечує можливість впровадження та використання комплексу в будь-якій виробничій мануфактурі та встановлення майже на будь-якому підприємстві. Окрім розміщення, такий комплекс дозволяє розробити гнучку систему загрузки матеріалу в нього, це може буде пряма загрузка з виробничої лінії (саме такий комплекс розглядається в роботі) та загрузка з зовнішніх джерел, такі як машини, мішки, тощо.

Комплекс забезпечує автоматичну подачу та забезпечення матеріалом подальших споживачів, а також може автоматично забезпечити проходження матеріалу з ємностей зберігання до кінцевого споживача, в залежності від вибраних налаштувань. Кінцевим споживачем в такому випадку може бути прийомна ємність іншої виробничої лінії, як в випадку розглянутому в роботі, так і в машини, вагони.

Такий комплекс забезпечує оптимальне використання складських приміщень, що сприяє росту економіки підприємства. Комплекс також може бути налаштований під різні специфічні вимоги замовника. Комплекс представлений в роботі має в свої конструкції металічний сепаратор, для відокремлення метали з сировини.

Основним елементом безтарного сховища речовини є силос рисунок 1.1.

Силос – металічна ємність переважно циліндричної форми з конусом

внизу. Основне призначення силоса – зберігання сипучих речовин, таких як зерно, посипки, пісок, крейда. Великою превагою силоса є те, що така конструкція не займає багато виробничих площ, які у свою чергу можуть значно збільшити промислові витрати, та дозволяє спроектувати та розробити силос під потребу споживача, тобто підібрати потрібну займану площу та ємність силос, згідно с роботою[1].

Рисунок 1.1 – Силос

Комунікації між силосами можуть бути організовані у разні способи. Для зерна, піску, посипки переважно використовують різного виду транспортери, для рідин використовують труби. Комплекс представлений у роботі здійснює комунікацію за допомогою труб, а транспортування флекси відбувається за допомогою повітря, яке подається в ці труби за допомогою компресора високого тиску рисунок 1.2.

Рисунок 1.2 – Компресор

Для регулювання потоку матеріалу та вибору напрямку його руху використовується перемикач потоків (дивертор) рисунок 1.3.

Рисунок 1.3 – Дивертор(перемикач потоків)

Дивертор – пристрій, який дозволяє регулювати напрям руху речовини по трубах. Пристрій має один вхід та два, або більше виходів між якими і відбувається переключення потоку. В комплексі використовують пневматичні дивертори, які використовують повітря для переключення потоків, також бувають механічні дивертори, перемикання в яких відбувається оператором.

Схожим на дивертор елементом комплексу є заслінки, за тим виключення що заслінку не перемикають поток, а зупиняють його повністю. Для контролю положення цих елементів, тобто контроль напрямку дивертора та закриття чи відкриття заслінок використовують кінцевики рисунок 1.4.

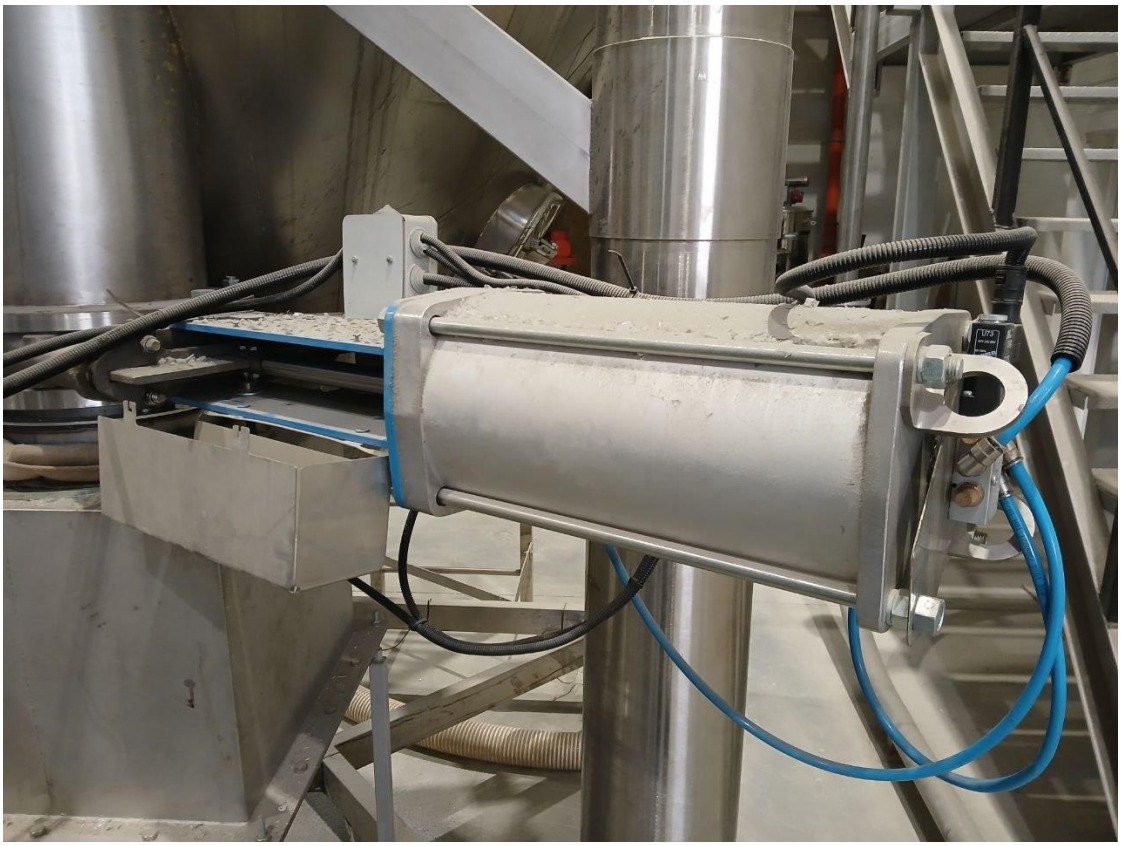


Рисунок 1.4 – Пневматична заслінка

Для усунення можливості попадання металу к кінцевому споживачу, в комплексі встановлений метало-сепаратор рисунок 1.5. Він представляє собою заслінку та метало-детектор перед нею, якщо в потоці був помічений метал, то

заслінка спрацьовує та на деякий час перемикає потік в спеціальну ємність з сировиною з металом.

Рисунок 1.5 – Метало-сепаратор

1.2 Основні задачі та цілі комплексу

Комплекс безтарного сховища сировини виконує декілька основних задач для взаємодії двох зв’язаних виробничих ліній.

Перша задача є ефективне використання виробничіх приміщень та площі. Ні одне виробництво не хоче займати площу для простого зберігання сировини, адже набагато ефективніше відразу подавати сировину в виробництво, та не витрачати на неї місце. Комплекс дозволяє використовувати мінімальну площу для зберігання сировини, а також дозволяє розмістити основні зберігальні ємності в місцях, які не підходять для інших задач, говориться в джерелі [2].

Друга задача – забезпечення зв’язку без участі людини між двома лініями, де одна лінія забезпечує сировиною іншу. В нашому випадку лінія А виробляє пет-флексу для лінії Б, яка використовую флексу, як сировину. Задача першої лінії є наповнення силосів в яких і зберігається флекса, потім по запиту другою лінії комплекс повинен забезпечити подання флекси. Основною перевагою такого підходу є декомпозиція, тобто перша лінія виробляє сировину в не залежності від другою лінії, завдяки присутності комплексу, який і контролює взаємодію.

Комплекс забезпечує безліч додаткових рішень в залежності від потреб замовника. Такі рішення можуть включати додаткову обробку, сортування, сушку сировини перед потраплянням у фінальну лінію. В представленому комплексі присутні два основних силоса для зберігання двох видів сировини, яка в подальшому подається в міксер, згідно рецепту, де відбувається перемішування для подальшої подачі міксу. Сховище обладнане метало-детекторами, які відокремлюють небажанні елементі під час транспортування.

Отже комплекс забезпечує зв’язок між вузлами виробничого концерну, який в свою чергу потребує мінімального втручання людини, забезпечує гнучкість та можливість адаптування під задачі замовника, впровадження додаткових вузлів. Все це відбувається на відносно малій площі, завдяки чому відбувається економія виробничих приміщень, джерело [3].

1.3 Компонентна база комплексу

Керування комплексу здійснюється в автоматичному режимі від початку і до кінця, за виключенням вибору режиму роботи комплексу оператором в залежності від поточної задачі. Оператор встановлює основні налаштування комплексу такі як:

– вибір джерела завантаження. Це може бути лінія з виготовлення продукції чи зовнішнє джерело;

– рецепт приготування продукції, якщо присутній більш ніж 1 вид сировини;

– кінцевий споживач сировини. В випадку представленому в проекті може бути виробнича лінія або вивантаження сировини в ємності для подальшої реалізації.

Центральним апаратним забезпеченням є контролер серії Siemens S7-1200 рисунок 1.6.

Рисунок 1.6 – контролер Siemens S7-1200

Це багатофункціональний контролер, який дозволяє об’єднати в собі майже всі елементи периферії, виконати циклічну програму роботи та взаємодіяти з елементами людино-машинного інтерфейсу.

Обраний контролер забезпечує високу продуктивність та відмовостійкість,

що дозволяє розробляти програми майже будь-якої складності з досить складними алгоритмами та багатою архітектурою.

Контролер забезпечує можливість встановлення до восьми пристроїв периферії для роботи з дисктретними, аналоговими, аналоговими-температурними входами-виходами, також підтримує протоколи обміну інформацією такі як Ethernet, Modbus, Profinet, RS-485.

Керування комплексом здійснюється за допомогою керування як і дискретними сигналами так і аналоговими, відповідно до джерела[4].

Для взаємодії з дискретними сигналами використовується модуль SM 1223 DC рисунок 1.7.

Рисунок 1.7 – Дискретний модуль SM 1223 DC

Цей модуль представляє собою набір з 16 дискретних входів та 16

дискретних виходів. Здебільшого цей модуль приймає сигнали з датчиків рівня, кінцевих вимикачів, стан контактора, а його дискретні виходи вмикають реле для старту частотних перетворювачів.

Для взаємодії з аналоговими сигналами, такими як: ультразвукові датчики, швидкість з частотних перетворювачів, використовується модуль аналогових входів-виходів SM 1231 AI рисунок 1.8.

Рисунок 1.8 – Аналоговий модуль SM 1231 AI

Взаємодія оператора з комплексом відбувається за допомогою Siemens HMI рисунок 1.9.

Рисунок 1.9 – Siemens HMI

Це спеціальна панель яка дозволяє розробити інтерфейс для взаємодії з комплексом, який включає в себе:

– вибір основних налаштувань комплексу та режиму роботи;

– керування в ручному режимі, за потреби виходу з нештатних ситуацій або усунення виробничих помилок;

– тестовий режим комплексу для налагодження коректної роботи та підбір основних параметрів;

– відображення стану комплексу.

Панель розміщується в будь-якому зручному місці за потребою замовника, так як з’єднується з контролером за допомогою інтернет кабелю. Панель має вбудовану файлову систему, що дозволяє вести додаткові можливості, такі як бухгалтерський облік за потреби, згідно джерела [5].

Основними датчиками периферії є датчики рівня, які встановлюються

майже в кожний силос, саме вони дозволяють контролювати поточний стан силоса та запобігти пересипанню рисунок 1.10.

Рисунок 1.10 – Аварійний датчик рівня

Для деяких силосів потрібні більш точні системи контролю наповнення, для таких цілей використовуються ультра-звукові датчики з HART протоколом рисунок 1.11. Цей протокол дозволяє передавати дані та живити датчик по одному кабелю, що значно знижує витрати.

Рисунок 1.11 – Ультразвуковий датчик рівня

Як було сказано раніше, в комплексі присутня спеціальна ємність – міксер. Для точного приготування рецепту сировини, тобто змішування в правильних пропорціях, використовуються тензодатчики рисунок 1.12.

Рисунок 1.12 – Тензодатчик

Такі датчики перетворюють механічне навантаження на них в аналоговий сигнал 0-10V, 0-20mA, 4-20mA, який потім обробляється спеціальним модулем Siwarex WP231 рисунок 1.13.

Рисунок 1.13 – Ваговий модуль Siwarex WP231

Після обробки цим модулем, значення ваги йде в основний ПЛК, де з ним може взаємодіяти програма.

Керування моторів здебільшого відбувається за допомогою частотних перетворювачів рисунок 1.14.

Рисунок 1.14 – Частотний перетворювач ABB

Це пристрої, які змінюють частоту вхідного перемінного струму, тим самим дозволяючи регулювати швидкість обертання трифазного асинхронного двигуна, які використовуються у компресорах, мішалках, шнеках. Економія електроенергії ще одна велика перевага частотних перетворювачів, завдяки можливості прибрати зайве навантаження на двигун і більш точно контролювати його обертання.

Більшість сучасних частотних перетворювачів мають великий інтерфейс взаємодії з іншими пристроями такі як контролери. Вони можуть керуватися за допомогою як і «сухих контактів», так і протоколів Ethernet, Modbus. Для контролю поточного стану двигуна по тим самим протоколам можно отримувати інформацію про швидкість двигуна, струм, навантаження, попередження, помилки, напрям руху, тощо.

Система безпеки та контролю реалізовані за допомогою індуктивних датчиків, які використовуються для підтвердження положення заслінок та диверторів рисунок 1.15.

Рисунок 1.15 – Індуктивний датчик

Вся периферія використана в комплексі може бути замінена за потреби на іншу, що забезпечує гнучкість такого комплексу та можливість адаптації під будь-які потреби.

1.4 Постановка задачі

Розглянувши компонентну базу комплексу та його основні задачі, можемо зробити висновки, що комплекс забезпечує зручну та практичну роботу з сипучою сировиною та має гнучку компонентну базу, яка може змінюватися під потреби замовника.

Комплекс має в своєму складі багато виконавчих механізмів таких як: двигуни, дивертори, заслінки, тощо. Для керування всіма цими механізмами слід розробити програму, як для кожного окремого механізму так і для автоматичної роботи комплексу.

Система також включає в собі багато датчиків, такі як ультра-звукові та тензодатчики, аналогові сигнали яких слід правильно обробляти для подальшого використання в програмі.

Для зручної взаємодії з комплексом, розробити людино-машинний інтерфейс, який має бути інтуїтивно зрозумілий та прозорий для оператора. Інтерфейс має відображати основні параметри та поточний стан комплексу, керування в ручному режимі та налаштування параметрів.

Одним з головних вимог до будь-якою промислової системи є надійність. Слід розрахувати цей параметр для комплексу, для розуміння його користі, адже часті відмови призводять до збитків підприємства.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСУ

2.1 Етапи роботи комплексу

Алгоритм роботи комплексу в основному полягає в циклічному опитуванні поточного стану комплексу та наповненості силосів, після чого, при сигналах на завантаження чи вивантаження, створення правильної конфігурації комплексу для правильного транспортування матеріалу через нього.

Для точного розуміння порядку роботи комплексу нижче приведена принципова схема комплексу рисунок 2.1.

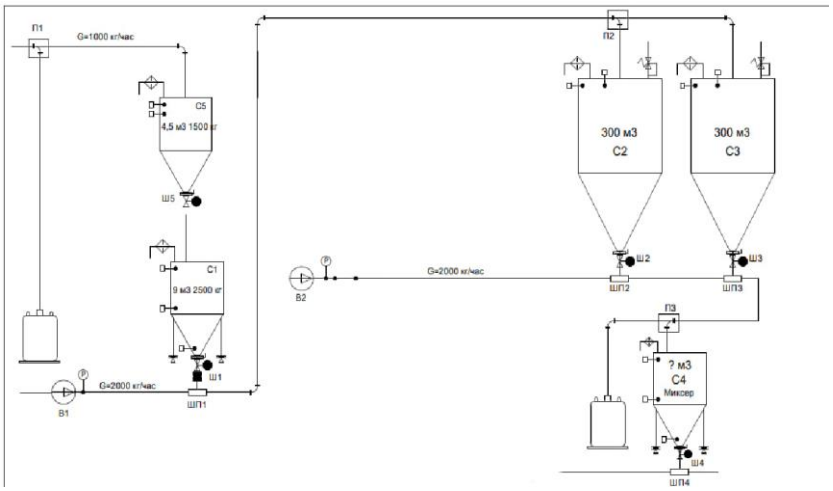


Рисунок 2.1 – Принципова схема комплексу

На принциповій схемі можно побачити два основних накопичувальних силоса С2 та С3, які зберігають основну частину сировини. Завантаження відбувається через ємність С1, яка завдяки тензодатчикам, відміряє чіткі порції сировини для подальшого завантаження в С2 та С3. Ємність С5 використовується у якості «буферної зони», коли з ємності С1 йде вивантаження, С5 накопичує сировину з першої лінії-донора, у роботах [6] – [7].

Після С2 та С3 встановлений дивертор П3, який дозволяє обирати напрямок завантаження сировини, це можуть бути мішки, або міксер С4. Міксер у свою чергу підтримує можливість створення рецепту, завдяки якому можна

комбінувати різні сировини з С2 та С3. Останній етап, на запит другої лінії міксер починає вивантаження сировини.

Основні етапи операторської роботи:

– оператор обирає режим завантаження сировини у мішки або у ємності С2 та С3, якщо були обрані ємності С2 та С3 оператор обирає вагу кожної порції та ємність в яку буде відбуватися завантаження;

– оператор задає режим вивантаження сировини з ємностей С2 та С3. Це може бути вивантаження в мішки або в міксер С4. Якщо обрано режим вивантаження в С4, оператор завдає рецепт у вигляді ваги сировини з кожної ємності.

Основні етапи програмної роботи:

– програма перемикає дивертор П1 в залежності від обраного режиму, перемикає дивертор П2 в залежності від обраної ємності та перевіряє чи датчик аварійного рівня ємності, якщо датчик спрацьований програма інформує про це, відкриває заслінку Ш5;

– після початку процесу завантаження програма чекає набрання С1 до відповідної ваги, після чого закриває заслінку Ш5, відкриває заслінку Ш1 та вмикає компресор В1, починаючи процес вивантаження в ємності;

– при досягненні нульової ваги, зачинається заслінка Ш1 та вимикається В1 і повторюється пункт 1;

– для вивантаження програма перемикає дивертор П3 у відповідне положення, перевіряє датчик аварійного рівня та вмикає компресор В2;

– в залежності від рецепту програма відчиняє заслінку Ш2 та набирає необхідну вагу з ємності С2, після чого зачиняє Ш2, відчиняє Ш3 та набирає відповідну вагу з С3, після чого зачиняє Ш3;

– по подальшому запиту, програма відкриває Ш4 та починає процес перекачки сировини з міксера в лінію, при досягненні нульової ваги, програма виконує пункт 4.

Таким чином при завданні первинних операторських налаштувань

програма в автоматичному режимі забезпечує неперервну подачу сировини, контролюючи при цьому завантаженість комплексу. Налаштування у свою чергу можно змінити майже в будь-яку мить, що не нашкодить роботі комплексу, адже система побудована децентралізовано, що означає кожен блок працю автономну зі зв’язком з іншими блоками, а отже зміна стану одного не завжди призводить до зміни всієї системи.

2.2 Програма керування двигунами

Виконання програми виконує контролер Siemens S7-1200, до якого приєднанні модулі цифрових входів-виходів для взаємодії з цифровими датчиками, керування пускачами та керування сигнальними колонами та модулі аналогових входів- виходів для взаємодії з аналоговими датчиками, такими як ультразвукові, та керування швидкістю двигунів, джерело [8].

Програмування контролеру відбувається в програмному забезпечені TIA Portal від компанії Siemens. Це середа розробки для забезпечення систем автоматизації технологічних процесів, яка відповідає за повний комплекс автоматизації від логіки програми до людино-машинного інтерфейсу. Середа розробки використовую декілька мов програмування такі як: FBS, KOP, STL.

Перший етап програмування полягає у створенні конфігурації майбутньої системи, тобто обрання контролеру, периферії, HMI-панелей та конфігурування інтернет з’єднань та інших протоколів. Система представлена у проекті має наступну конфігурацію рисунок 2.2.

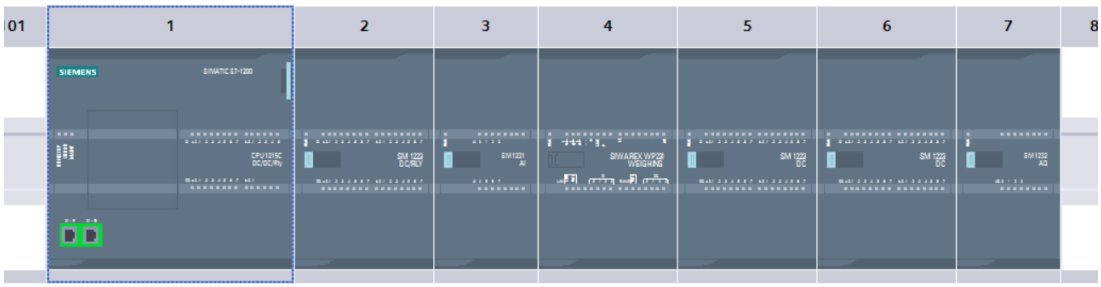


Рисунок 2.2 – Конфігурація контролеру

Основними модулями периферії є модуль SM1223 – модуль цифрових входів-виходів, SM 1231 – модуль аналогових входів та модуль SM1232 – модуль аналогових виходів. Спеціальний модуль під номер чотири – це Siwarex WP231. Це модуль призначений для взаємодії з тензодатчиками, обробці їх сигналів та отримання ваги рисунок 2.3.

Рисунок 2.3 – Siwarex WP231

Другий етап є створення конфігурації зв’язку, тобто проведення основних комунікаційних протоколів та додавання всіх учасників в проект. В даному випадку, контролер програми за допомогою PROFINET з’єднується з HMI та іншим контролером першої лінії, для отримання інформації про поточний стан лінії. Повна конфігурація представлена на рисунку 2.4.

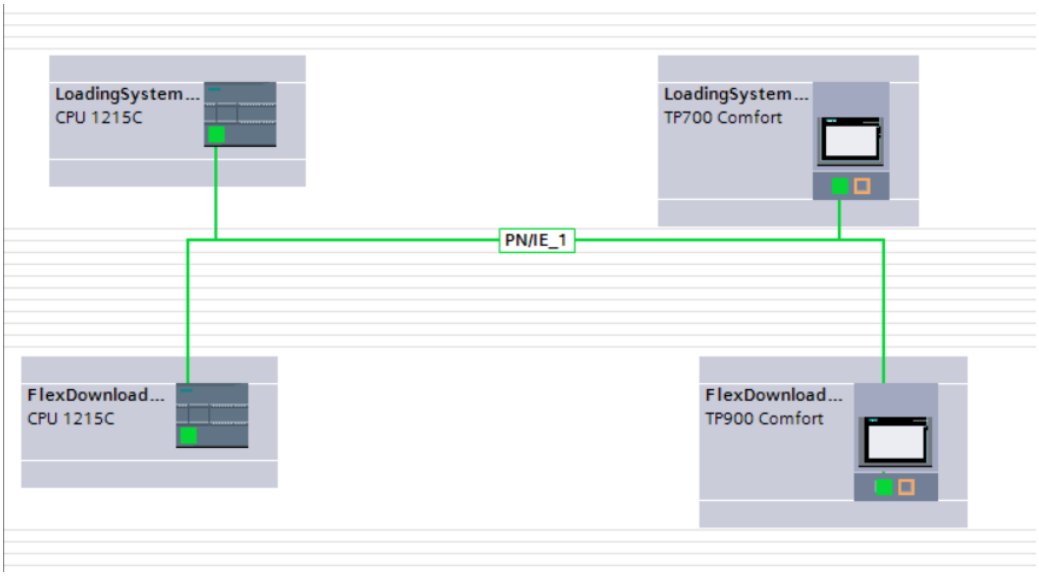


Рисунок 2.4 – Повна конфігурація проекту

Третім етапом слід назначити зміні входам та виходам, вказавши їх адреса. Адреса входів виходів назначаються автоматично, але є можливість вказати їх вручну. Індекс I означає цифровий вхід, такий вхід займає в контролері один біт пам’яті та може зберігати два значення 0 чи 1, на рисунку 2.5 приведено приклад визначення цифрового входу, як у роботі [9].

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	AlarmM1	Bool	%I0.0	☐	☑	☑	☑	

Рисунок 2.5 – Визначення цифрового входу

На рисунку 2.5 можно побачити у першій колонці ім’я змінною, за яким в подальшому в програмі ми будемо звертатися до цієї адреси. AlarmM1 – зміна яка зберігає інформація про наявність помилку двигуна M1, якщо значення перемінної 1 – двигун має помилку, якщо 0 то двигун готовий до роботи.

Колонка «Data type» – визначає тип даних змінною, в даному випадку тип даних є «Bool» – тип даних який може зберігати значення від 0 до 1, або як часто прийнято називати TRUE або FALSE.

Колонка «Address» – вказує на місце в пам’яті в якому зберігається змінна. Перша цифра адреси вказує на номер байту, друга цифра після крапки на номер

біту де зберігається остаточне значення.

Колонка «Retain» несе функція зберігання стану змінної після зняття живлення з контролеру, але в даному випадку колонка не активна, так як стан цієї змінною визначається зовнішнім фактором.

Колонка «Accessible from HMI/OPC UA/Web API» визначає можливість прямого зчитування стану змінної для HMI панелі або Web интерфейсу під час роботи програми.

Колонка «Writable from HMI/OPC UA/Web API» визначає можливість прямого редагування стану змінної та її параметрів для HMI панелі або Web интерфейсу.

Колонка «Visible in HMI engineering» визначає можливість мониторингу стану змінної HMI.

За тим самим принципом визначається і аналогові входи. На рисунку 2.6 можно побачити визначення аналогового сигналу поточної частоти для моторів.

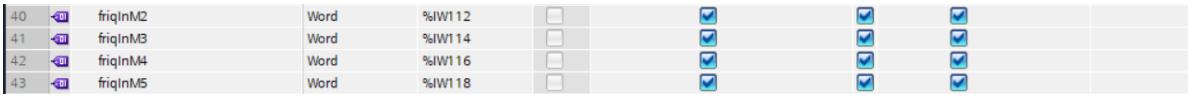


Рисунок 2.6 – Аналогові входи

Основна відмінність полягає у новій адресі зберігання. Індекс IW означає що зберігається змінна яка займає 2 байти пам’яті тобто 16 біт. Перша цифра означає початок запису змінної в пам’яті, так IW12 означає що, зміна займає 2 байти пам’яті починаючи з 12 байту включно.

Тип даних у даному випад «Word». Цей тип даних, як було сказано раніше займає 16 біт пам’яті та здатен зберігати наступні значення такі як: Integers, Binary numbers, Octal numbers, Hexadecimal numbers, BCD, Decimal sequence.

Цифрові виходи мають ті самі параметри за тим лиш виключенням, що їх зміна відбувається не з зовні а в середині програми та їх індекс позначається літерою Q. На рисунку 2.7 приведено приклад визначення цифрового виходу старту двигуна.

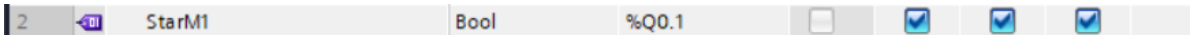


Рисунок 2.7 – Цифровий вихід старту двигуна

Коли програма змінює значення змінної «startM1» на 1, на виході з’являється напруга 24V, яка і вмикає реле для старту двигуна.

За аналогічним принципом визначаються аналогові виходи рисунок 2.8.

Рисунок 2.8 – Аналогові вихідні значення частот двигунів

Слід зазначити, що деякі модулі підтримують різні варіанти завдання аналогових входів виходів. Це може бути 0-20mA, 4-20mA, 0-10V. Побачити налаштування аналогового каналу QW160 можно на рисунку 2.9.

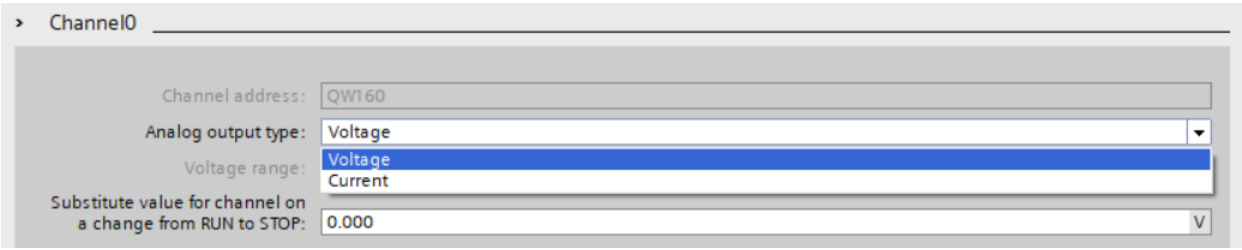


Рисунок 2.9 – Налаштування аналогового каналу

Як можно побачити модуль підтримує можливість перетворення значення змінної у вигляді вихідної напруги або струму. В режимі напруги (0-10V) максимальному значенню змінної буде відповідати 10V напруги на аналоговому каналі відносно 0V контролеру, а в режимі струму, максимальному значенню змінної буде відповідати вихід 20mA.

Важливою можливістю також є створення власних змінних у внутрішній пам’яті контролеру, які не прив’язані до фізичних входів чи виходів. Створення

власної змінною приведено на рисунку 2.10.

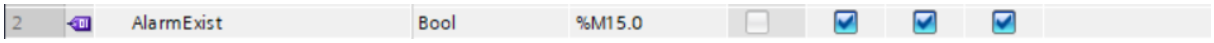


Рисунок 2.10 – Створення змінної у внутрішній пам’яті контролеру

В даному випадку індекс М означає що ми використовуємо внутрішню пам’ять контролеру, а цифру вказують на номер байту та біту в пам’яті.

Слід зазначити що існують випадки коли існує можливість взаємодії двох різних змінних з однією областю пам’яті, що призводить до подальших конфліктів. На рисунку 2.11 приведено приклад конфлікту двох змінних.

Рисунок 2.11 – Конфлікт двох змінних

В даному випадку маємо наступний конфлікт. Змінна 2 займає 16 біт пам’яті, тобто 2 байти, починаючи з 110 байту. В той самий час Змінна 1 займає 5 біт 110 байту, що означає, що якщо Змінна 2 записала у 5 біт значення 0, після чого далі в програмі відбулася ситуація, при якій Змінна 1 була змінена на 1 то у 5 біті з’явиться одиниця, яка змінить значення числа Змінної 2, що призведе до подальших помилок. Отже слід бути дуже уважним при роботі з пам’яттю та не допускати такого роду накладання.

Також існують зони пам’яті, які зайняті самим контролером для надання користувачеві корисних стандартних функцій. На рисунку 2.12 приведено приклад таких змінних.

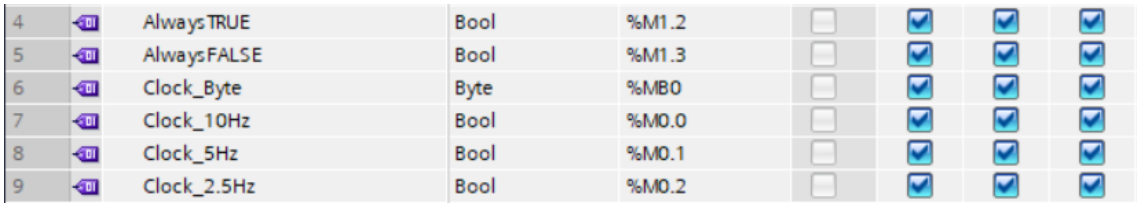


Рисунок 2.12 – Системні змінні контролеру

Змінна «AlwaysTRUE» – змінна яка завжди має значення TRUE, тобто зберігає в собі завжди значення 1. Ця зміна може бути корисною для ввімкнення чи вимкнення певних функцій системи на рівні коду, чи блокування певних датчиків чи сигналів. Функція «Clock_10Hz» – системний годинник, що видає високий імпульс 10 разів на секунду. Може бути корисним для підрахунку певних процесів, або включення певних процесів 10 разів на секунду.

Програма комплексу переважно складається з блоків, які називаються функціональні блоки. Функціональні блоки це набір коду для кожного окремого вузла починаючи з низу ієрархії, тобто просто механізми такі як двигуни, дивертори, заслінки, які потім складаються в блок який включає логіку роботи цих окремих блоків.

Присутня можливість створювати свої складні типи даних, аналог «класів» у більшості мовах програмування. Такі складні типи даних дозволяють зробити аналог фізичних систем у програмі для легшої роботи з ними.

Основним механізмом для роботи системи є двигуни, які використовуються у багатьох вузлах системи. Для початку створимо тип даних «Motor» який буде зберігати поточний стан двигуна та мати наступні властивості:

– «Start» – зміна, яка показує чи є команда на старт двигуна в поточний час;

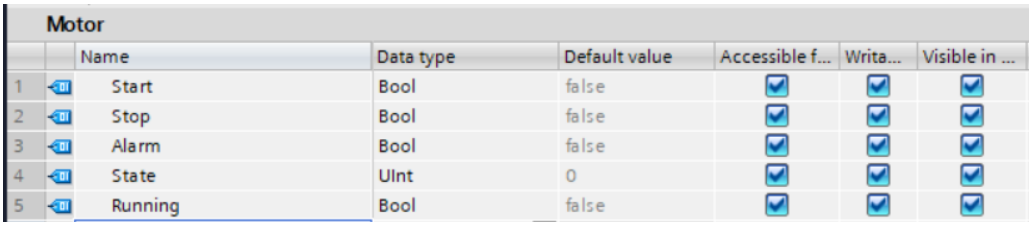
– «Stop» – зміна, яка вказує на те, що двигун перебуває в стані зупинки;

– «Alarm» – помилка двигуна, визначає чи має двигун помилки;

– «Running» – зміна, яка вказує на стан двигну в якому він набрав поточні обороти та працює в нормальному режимі;

– «State» – загальна зміна стану, на відміну від інших змінних, тип даних цієї змінної є Integer, вона зберігає 4 стани двигуна, 0 – стоп, 1 – розгін, 2 – робота, 3 – помилка відповідно.

На рисунку 2.13 можно побачити приклад даного типу даних



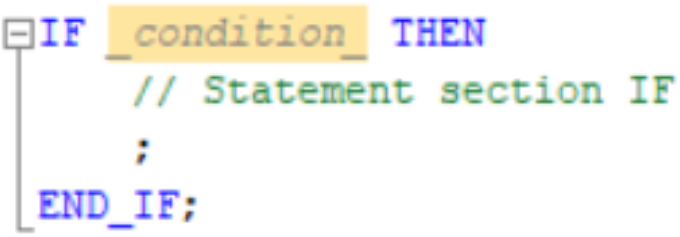
Motor

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...
1	Start	Bool	false	☑	☑	☑
2	Stop	Bool	false	☑	☑	☑
3	Alarm	Bool	false	☑	☑	☑
4	State	UInt	0	☑	☑	☑
5	Running	Bool	false	☑	☑	☑

Рисунок 2.13 – Тип даних «Motor»

Після створення типу даних створюємо функціональний блок «SimpleConvertorsFB», який буде виконувати логіку роботи двигуна.

Основним інструментом цього блоку є інструкція «IF». Ця інструкція приймає в себе булеві аргументи та на основі результату булевого рівняння виконую чи не виконує код в середині блоку. Приклад інструкції «IF» приведено на рисунку 2.14.



```
IF _condition_ THEN
    // Statement section IF
    ;
END_IF;
```

Рисунок 2.14 – Інструкція IF

Замість «_condition_» вставляється булевий вислів, який при значенні 1 виконує код між ключовими словами «THEN» та «END_IF», в іншому випадку оминає цей сегмент.

Перш за все створюємо зміні, з якими буде працювати наш функціональний блок. В нього будуть входити наступні зміні, джерело[10].

Зміні на вхід:

– «Alarm» – зовнішня помилка двигуна;

– «FriqIn» – вхідна частота двигун, тобто його швидкість обертання;

– «FriqUst» – задана частота обертання двигуна.

Зміні на вихід:

– «Main» – команда старту двигуна, яка включає фізичний вихід на високий сигнал;

– «FriqOut» – вихідне завдання для інвертора, визначає частоту обертань двигуна;

– «FriqHMI» – вихідна частота для відображення швидкості на HMI.

Зміні на вхід та вихід, тобто зміні які можуть редагуватися зовні і в середині:

– «Motor» – спеціальний тип даних, який ми створили раніше;

– «Start» – команда старту двигуна.

Статичні зміні:

– «SCALE_Instance» – масштабування вхідного сигналу частоти;

– «DeSCALE_Instance» – демаштабування вихідного сигналу

Допоміжні зміні:

– «Temp_FriqIn» – тимчасова зміна для вхідної частоти;

– «Temp_FriqOut» – тимчасова зміна для вихідної частоти;

– «Temp_FriqReal» – тимчасова зміна для частоти, яка відображається на HMI.

Всі зміні блоку можно побачити на рисунку 2.15.

SimpleConverterFB

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...
1	Input				☐	☐	☐
2	Alarm	Bool	false	Non-retain	☑	☑	☑
3	FriqIn	UInt	0	Non-retain	☑	☑	☑
4	FiriqUst	Int	0	Non-retain	☑	☑	☑
5	Output				☐	☐	☐
6	Main	Bool	false	Non-retain	☑	☑	☑
7	FriqOut	UInt	0	Non-retain	☑	☑	☑
8	FriqHMI	Real	0.0	Non-retain	☑	☑	☑
9	InOut				☐	☐	☐
10	Motor	"Motor"			☐	☐	☐
11	Start	Bool	false	Non-retain	☑	☑	☑
12	Static				☐	☐	☐
13	SCALE_Instance	"SCALE"			☑	☑	☑
14	DeSCALE_Instance	"DeSCALE"			☑	☑	☑
15	Temp				☐	☐	☐
16	Temp_FriqIn	Int			☐	☐	☐
17	Temp_FriqOut	Int			☐	☐	☐
18	Temp_FriqReal	Real			☐	☐	☐

Рисунок 2.15 – Зміні функціонального блоку мотор

Слід приділити увагу блокам «SCALE_Instance» та «DeSCALE_Instance». Це спеціальні блоки для обробки аналогових сигналів в значення з якими працює програма, тобто з 0-10V або 4-20 mA в Int або Real. Для запису аналогових сигналів в контролер драйвер перетворює значення сигналу в діапазоні від 0 до 27648, де значенню 0 відповідає найнижчій рівень сигналу, а значенню 27648 найвищій. На рисунку 2.16 приведено блок який обробки аналогового сигналу.

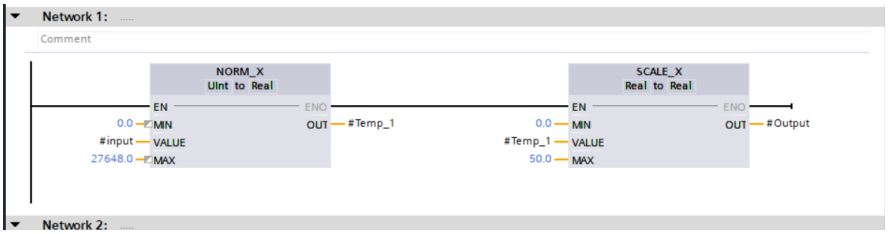


Рисунок 2.16 – Блок обробки аналогових сигналів

Перший етап – етап нормалізації значення. Вхідне значення нормалізується таким чином, щоб мінімальному значенню відповідала 0.0, а максимальному 1.0. Графік нормалізації приведено на рисунку 2.17.

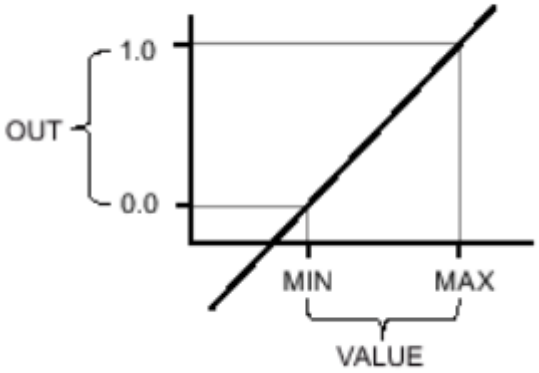


Рисунок 2.17 – Графік нормалізації величини

Фактично цей блок визначає відсоток вхідного значення від різниці між максимальним і мінімальним значеннями.

Другий блок – скалювання. Цей блок перетворює вихідне значення блоку нормалізації та записує його у вигляді величини між заданими межами, з якою програма працює в подальшому. На рисунку 2.18 приведено графік скалювання.

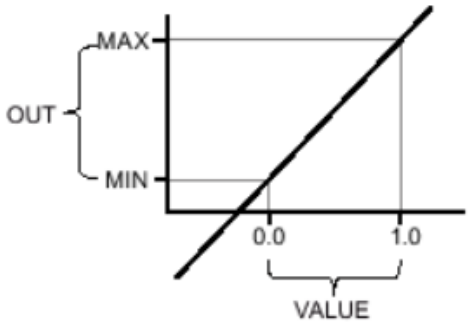


Рисунок 2.18 – Графік скалювання величини

Тобто перший блок визначає відсоток від шкали 0-27648, а другий від заданої шкали, в нашому випадку 0-50, що відповідає значенням частоти інверторів. Якщо значення вхідного сигналу буде дорівнювати половині від 27648, то вихідне значення буде дорівнювати 25 відповідно.

Перейдемо до функціонального блоку. На рисунку 2.19 приведено інструкцію IF для запуску двигуна, з джерела [10].

```
7  IF #Start AND NOT #Alarm AND #Motor.State <> 1 THEN
8      #Motor.Start := TRUE;
9      #Motor.Stop := FALSE;
10     #Motor.Running := FALSE;
11     #Motor.State := 1;
12     #Main := TRUE;
13 END_IF;
```

Рисунок 2.19 – Інструкція IF для запуску двигуна

В даному випадку для виконання запуску потрібно три умови. Перша це «Start» = 1, тобто наявність команди запуску. Друга це «Alarm» = 0, тобто

відсутність помилок. Третя це «Motor.State» <> 1, тобто двигун не знаходиться в стані розгону.

При виконанні всіх трьох умов оновлюється поточний стан двигуна та вмикається команда «Main», яка встановлює на відповідному виході контролеру логічну одиницю, яка запускає інвертор.

Після подання команди старт, слід дочекатися розгону мотора та вихід його на робочі обороти, за це відповідає інструкція, приведена на рисунку 2.20.

```
25 IF #Start AND (ABS(#FiriqUst - #Temp_FriqReal)<1) THEN
26     #Motor.State := 2;
27     #Motor.Running := TRUE;
28 END_IF;
```

Рисунок 2.20 – Інструкція перевірки роботи двигуна

При виконанні умови, що різниця між заданою частотою та реальною, по модулю, не перевищує 1, двигун вийшов на робочі обороти та працює відповідно, отже «Motor.State» = 2, що означає двигун знаходиться в роботі.

При виникненні помилки, перегрів, блокування, виконується інструкція приведена на рисунку 2.21.

```
37 IF #Alarm THEN
38     #Start := FALSE;
39     #Motor.Alarm := TRUE;
40     #Motor.Running := FALSE;
41     #Motor.Stop := TRUE;
42     #Motor.State := 3;
43 ELSE
44
45     #Motor.Alarm := FALSE;
46
47 END_IF;
```

Рисунок 2.21 – Інструкція помилки двигуна

Цей блок виконується при «Alaram» = 1, в такому випадку команда «Start» блокується та завжди міняється з 1 на 0, що не дає запуститися двигуну, а «Motor.State» = 3, що означає що двигун має аварію. При невиконанні умови Аварія двигуна знімається в «ELSE» інструкції.

Слід зазначити, що у цій інструкції не вимикається вихідна команда «Main», для її вимкнення існує інструкція приведена на рисунку 2.22.

```
15 IF NOT #Start THEN
16     #Motor.Start := FALSE;
17     #Motor.Stop := TRUE;
18     #Motor.Running := FALSE;
19     #Motor.State := 0;
20     #Main := FALSE;
21 END_IF;
```

Рисунок 2.22 – Інструкція вимкнення двигуна

Так саме у цій інструкція, яка виконується при умові «Start» = 0, з відповідного виходу контролеру знімається логічна 1. Ця інструкція працює при нормальному вимкненні двигуна та вимкненні при помилці. Слід зазначити, що при нормальному вимкненні при «Alaram» = 0, інструкція записує «Motor.State» = 0, що означає що двигун готовий к запуску, але при «Alaram» = 1, блок так само вимикає двигуна та записує «Motor.State» = 0, але далі виконається інструкція з рисунку 3.21, яка перезаписує стан двигуна як «Motor.State» = 3.

Отже ця інструкція виконається і в випадку нормальної зупинки та аварійної, але в будь-якому випадку записує правильний поточний стан.

Після написання функціонального блоку слід його визвати та назначити відповідні йому фізичні входи-виходи. Для цього створюється окремий блок в якому проводиться визначення таких сигналів для кожного окремого двигуна, адже ми створили функціональний блок, який працює к кожним двигуном такого типу, яких можу бути велика кількість.

На рисунку 2.23 приведено виклик функціонального блоку «SimpleConvertorsFB».

```
"SimpleConverterFB_DB_m4"(Alarm := "a_m4",
                          FriqIn := "friqInM4",
                          FiriqUst := "FriqUstConverters".UstM4,
                          Main => "m4",
                          FriqOut => "friqOutM4",
                          FriqHMI => "FriqHMI".UstM4,
                          Motor := "Motors".Motors[3],
                          Start := #StartM4);
```

Рисунок 2.23 – Виклик функціонального блоку «SimpleConvertorsFB»

В даному випадку визивається функціональний блок, який створює відповідну базу даних «SimpleConverterFB_DB_m4». Після виклику блоку треба задати йому відповідні зміні, з якими буде працювати наш блок:

– «a_m4» – фізичний вхід контролера, помилка двигуна;

– «friqInM4» – фізичний вхід 0 – 10V, швидкість обертання двигуна;

– «FriqUstConverters".UstM4» – установка швидкості двигуна з HMI панелі;

– «m4» – фізичний вихід контролера, який вмикає реле старту;

– «friqOutM4» – фізичний аналоговий сигнал 0-10V, завдання швидкості обертання двигуна;

– «FriqHMI".UstM4» – вихід швидкості двигуна для відображення на HMI панелі;

– «Motors".Motors[3]» – об’єкт класу «Motor», який зберігає поточний стан двигуна;

– «#StartM4» – команда старту двигуна з HMI панелі, або з програми.

Після виклику функціонального блоку та визначення відповідних змінних, слід викликати блок, який викликає в собі функціональні блоки «SimpleConvertorsFB». В даному проекті цей функціональний блок використовується для чотирьох різних моторів, виклик функціонального блоку приведено на рисунку 2.24.

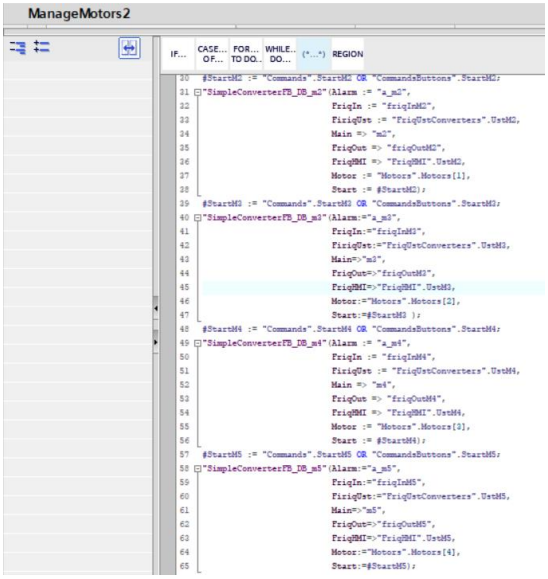


Рисунок 2.24 – Виклик функціонального блоку «SimpleConvertorsFB» для чотирьох моторів

Як можно побачити на рисунку 2.24 Виклик цих функціональних блоків відбувається в іншому блоці під назвою «ManageMotors2». Це допоміжний блок для зручності, який зберігає в собі всі виклики функціональних блоків пов’язаних з моторами.

Для того щоб програма почала працювати з мотором потрібно викликати блок ManageMotors2» в основному блоку програми «Main» або «OB1». Це основний блок програми, який і виконує її. Суть виконання полягає у циклічному виконані блоку «Main», за деякими виключеннями, з верху до низу.

Виклик функціонального блоку «ManageMotors2» у блоці «Main» приведено на рисунку 2.25.

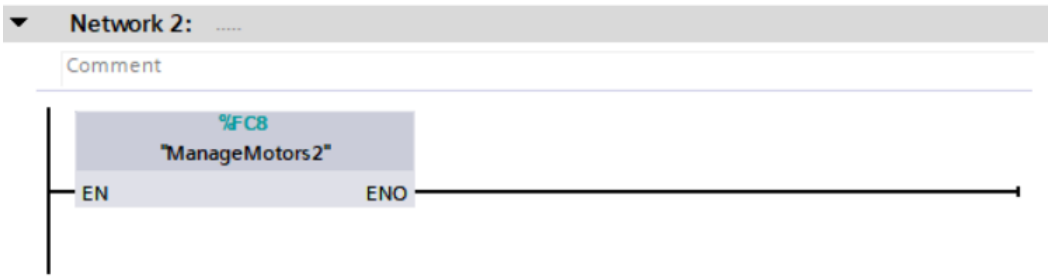


Рисунок 2.25 – Виклик функціонального блоку «ManageMotors2» у блоці «Main»

Повна архітектура з моменту отримання команди на старт двигуна до фактичного включення реле старту приведена на рисунку 2.26.

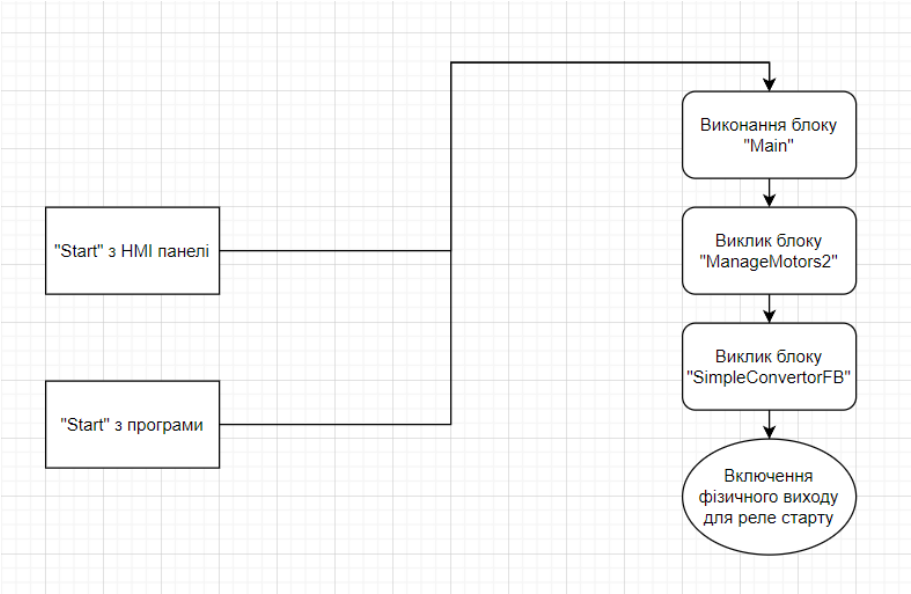


Рисунок 2.26 – ланцюг запуску двигуна

На першому етапі приходить команда «Start» з HMI панелі або з програми, для автоматичного режиму. На другому етапі виконується виклик функціональних блоків по ланцюжку. Після виклику функціонального блоку «SimpleConvertorsFB» виконується інструкція на рисунку 2.19 (інструкція старту). Після її виконання, на виході контролера (m4), з’являється логічна одиницю, яка вмикає реле старту двигуна.

2.3 Людино-машинний інтерфейс

Для взаємодії з контролером в проекті використовується HMI панель. Панель відображає поточний стан системи та можливість зміни основних параметрів системи, керування механізмами системи в ручному режимі.

TIA Portal має вбудовані інструменти для розроблення графічного інтерфейсу користувача. Інтерфейс включає в себе можливість малювання простих геометричних фігур, кнопок, слайдерів, списків, тексту, відображання стану змінних контролеру та відображання значення змінних, як зазначається в джерелі [11].

Головний екран розробленої програми включає в себе міксер для приготування рецепту сировини, мотори, компресор, кнопки пуску системи та окремих вузлів та відображення основних датчиків системи рисунок 2.27.

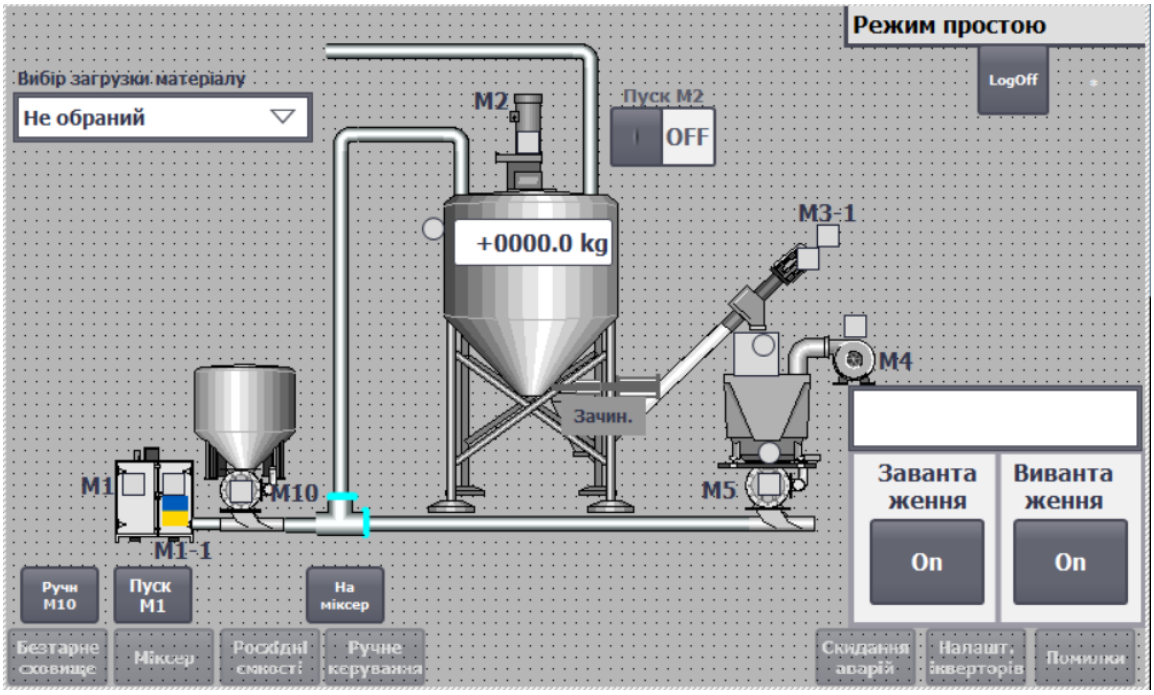


Рисунок 2.27 – Основний екран програми

Основний екран включає в себе особливий шаблон «Template_1», який відображається на всіх екранах HMI знизу та будуть накладатися зверху всіх інших елементів. Вони в основному використовуються для створення переходів

між сторінками на HMI панелі. Шаблон представлений на рисунку 2.28.

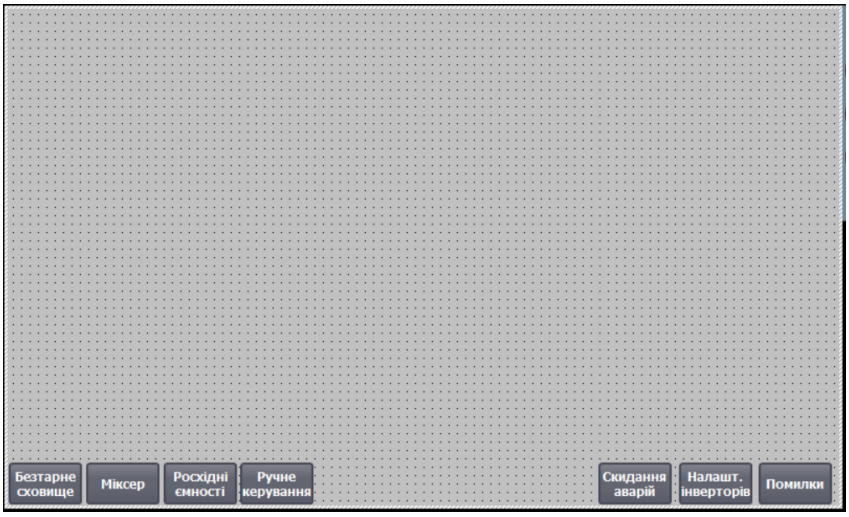


Рисунок 2.28 – Основний шаблон інтерфейсу

Після створення основного інтерфейсу слід створити спеціальні HMI – змінні, з якими буде взаємодіяти наш інтерфейс, а з ними в свою чергу будуть взаємодіяти змінні контролеру. Слід зазначити, що HMI панель має вбудовану систему, яка ніяк не зв’язана з контролером і може працювати окремо.

На рисунку 2.29 приведено приклад зв’язку між HMI змінною та змінною контролеру.

HMI tags

Name	Tag table	Da...	Connection	PLC name	PLC tag	...	Access mode
Commands_StartM1	Default tag table	Bool	HMI_Connection_1	LoadingSystemSpanbongPLC	Commands.StartM1		<symbolic access>

Рисунок 2.29 – Зв’язок між HMI змінною та змінною контролеру

Для зв’язку змінних слід вказати такі параметри:

– Name – ім’я змінною в HMI середовищі;

– Tag table – місце зберігання HMI змінною;

– Data type – тип даних цієї змінної;

– Connection – назва зв’язку між HMI та контролером, тобто який інтерфейс використовувати для обміну даними для цієї змінної;

– PLC name – назва ПЛК, змінна якого використовується;

– PLC tag – назва змінної, яка зберігається в ПЛК;

– Access mode – тип адресації до змінної. В даному випадку «symbolic access», тобто звернення за ім’ям змінної, також можно вказати абсолютну адресу змінної.

Після створення HMI змінної ми можемо використовувати її в інтерфейсі. В даному випадку змінна «Command_StartM1» дає команду на запуск двигуна M1, якщо змінна має значення 1, та відміняє пуск при значенні 0.

В середовищі TIA Portal у кожної кнопки є можливість визначити дію, яку вона буде виконувати при натисканні, утриманні, відпусканні. Основними такими функціями є:

– «InvertBit» – функція зміни стану змінної з 1 на 0, або з 0 на 1;

– «ResetBit» – функція встановлення значення змінної на 0;

– «SetBit» – функція встановлення значення змінної на 1.

Також важливою властивістю інструкція виконання цих функцій. Основними інструкціями є:

– «Click» – інструкція, яка виконується при одноразовому, короткочасному натисканні кнопки;

– «Press» – інструкція, яка виконується при тривалому натисканні кнопки;

– «Release» – інструкція, яка виконується при відпусканні кнопки.

Створимо інструкція для кнопки «Пуск M1» (рис 3.27), яка буде змінювати стан змінної «Command_StartM1». Для цього в властивостях кнопки створимо інструкцію «Click», яка буде виконувати команду «InvertBit» рисунок 2.30.

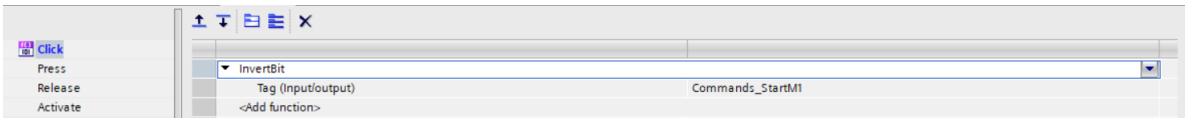


Рисунок 2.30 – Визначення функції при натисканні кнопки «Пуск M1»

Отже при натисканні кнопки, функція «InvertBit» буде встановлювати

протилежне значення в залежності від поточного для змінної «Command_StartM1», яка в свою чергу передає це значення змінній, яку ми вказали в зв’язку, як у джерелі [12].

Слід зазначити, що змінна «Command_StartM1» не тільки змінює значення змінної до якої прив’язана, а також отримує значення з змінної. Це означає, що якщо при натисканні кнопки, зміна залишається в поточному стані, програма блокує зміну стану, наприклад з причини помилки двигуна, яка вимикає команду старт рисунок 2.21.

Як зазначалося раніше HMI панель не тільки керує комплексом, а й відображає його стан та стан окремих вузлів. Для відображення стану двигунів використовується «Rectangle». На панелі малюється квадрат, який в залежності від стану двигуна буде змінювати свій колір тим самим повідомляючи про поточний стан двигуна рисунок 2.31.

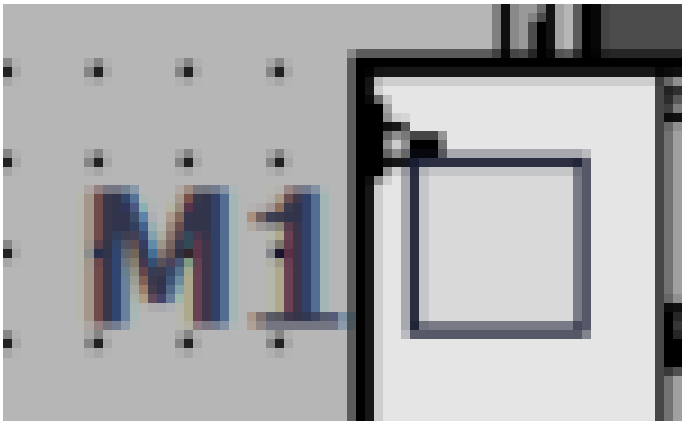


Рисунок 2.31 – Квадрат відображення стану двигуна

Цей квадрат має відповідні властивості, завдяки яким і можно відображати стан двигуна. Властивість яка використана в проекті є «Appearance». Вікно налаштування цієї властивості приведено на рисунку 2.32.

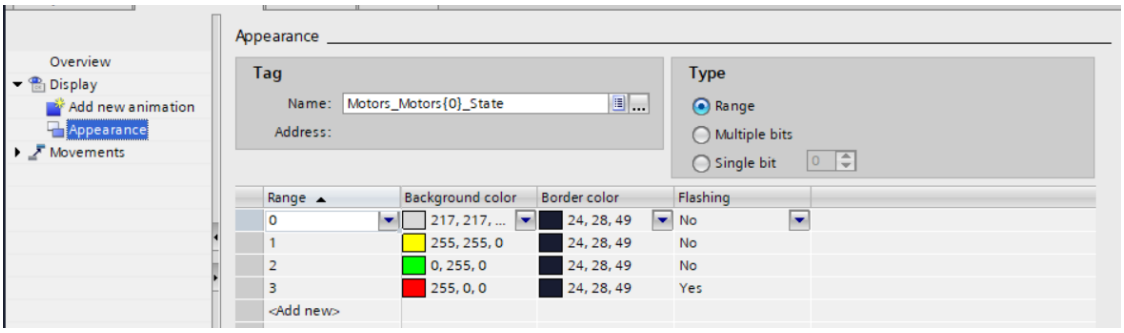


Рисунок 2.32 – Вікно властивості «Appearance»

Як і у випадку з кнопкою, перш за все потрібно вказати змінну, значення якою буде використовуватися для зміни параметрів квадрату. В даному випадку використовується зміна «Motors_Motors{0}_State», яка зберігає поточній стан двигуна. У таблиці нижче можно вибрати параметри квадрату в залежності від значення змінною, так можно побачити, що при значенні змінної 2, «Background color» встановлюється на зелений див рисунок 2.33, 2.34.

Motors	Array[0..15] of "Mo...	0.0			
Motors[0]	"Motor"	0.0			
Start	Bool	0.0	false	FALSE	TRUE
Stop	Bool	0.1	false	FALSE	FALSE
Alarm	Bool	0.2	false	FALSE	FALSE
State	UInt	2.0	0	0	2
Running	Bool	4.0	false	FALSE	TRUE

Рисунок 2.33 – Поточний стан двигуна

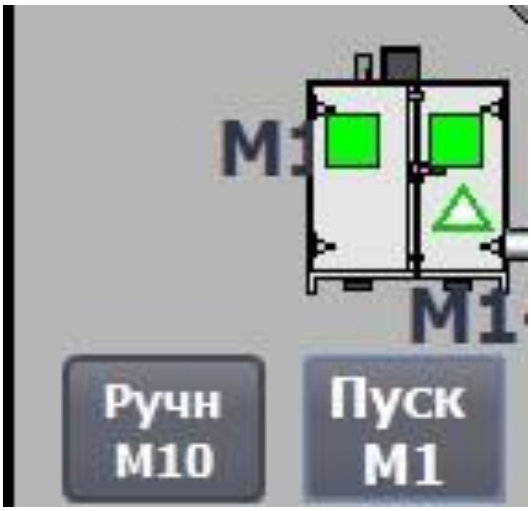


Рисунок 2.34 – Відображання поточного стану двигуна

2.4 Режим автоматичної роботи

Основною програмою комплексу є програма автоматичної роботи комплексу. Ця програма забезпечує безперервну подачу та прийом сировини, контроль за станом комплексу.

Для запуску комплексу в автоматичному режимі, оператор повинен обрати основні налаштування комплексу для подальшої роботи, а саме:

– обрати режим контролю компресора М1 рисунок 2.35;

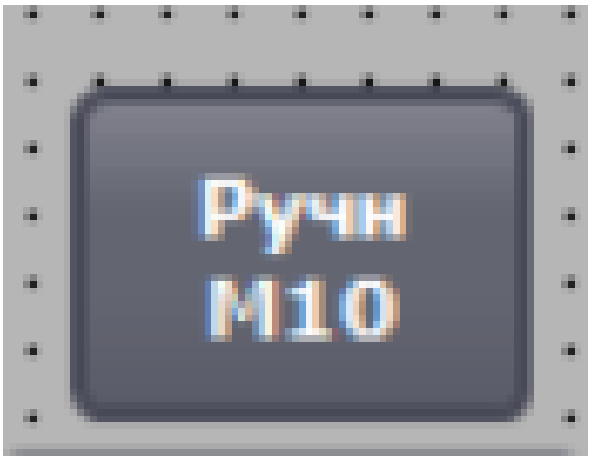


Рисунок 2.35 – Кнопка переведення компресора М1 в ручний або автоматичний режим

– Обрати спосіб завантаження міксеру зі списку на рисунку 2.36;

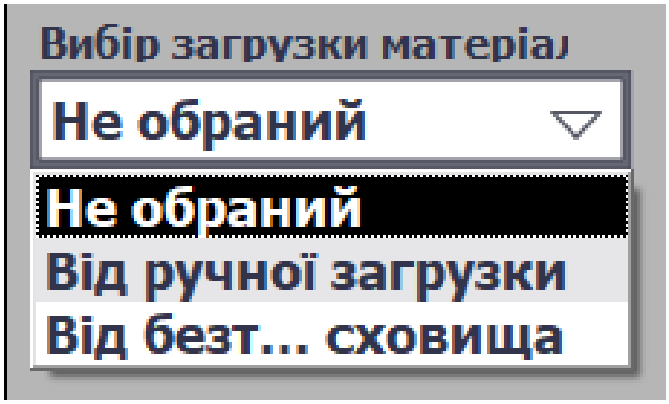


Рисунок 2.36 – Можливі джерела завантаження міксеру

– Задати швидкості обертання двигунів з інверторами рисунок 2.37;

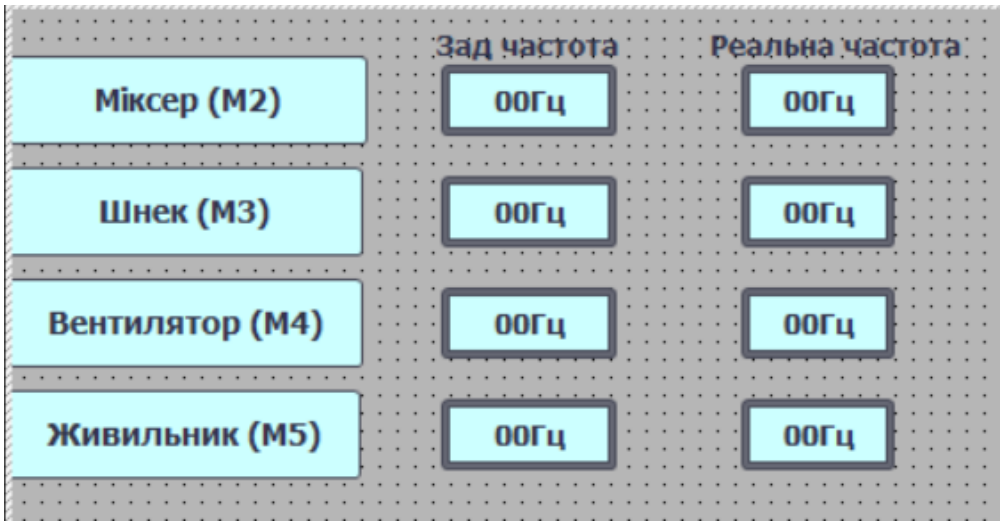


Рисунок 2.37 – Завдання для інверторів

Після налаштування комплексу оператор вмикає його на завантаження чи вивантаження міксеру однією з кнопок приведених на рисунку 2.38.

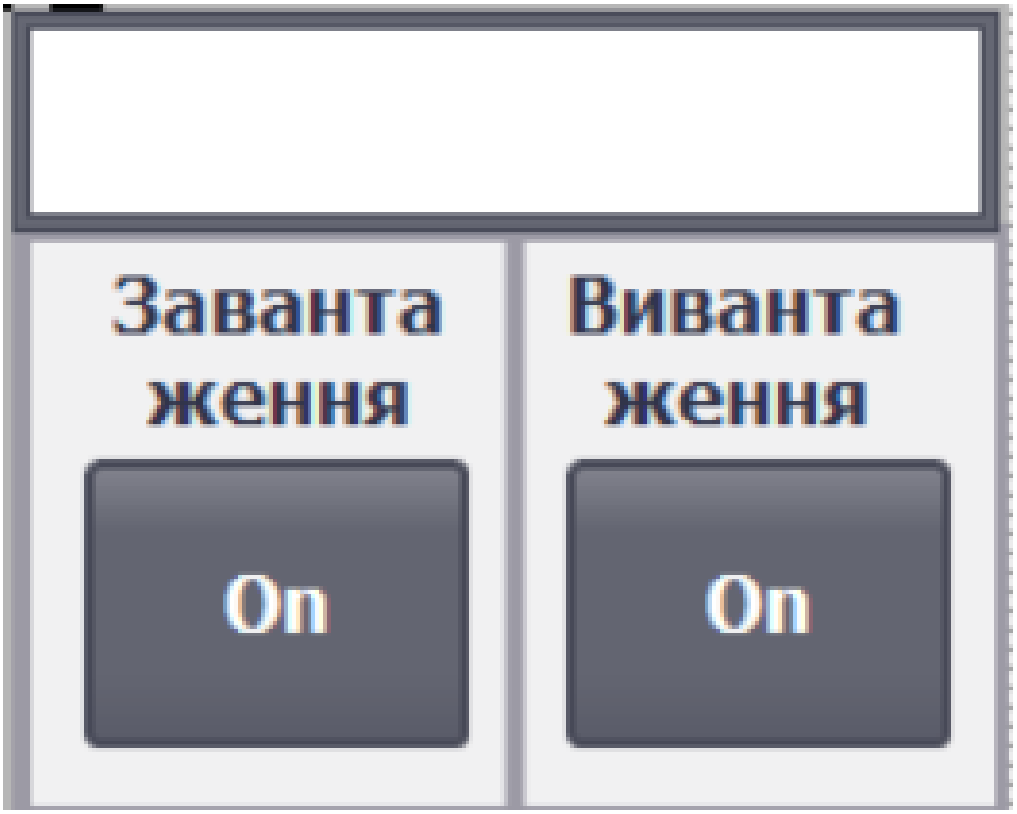


Рисунок 2.38 – Вибір режиму завантаження чи вивантаження міксеру

Після натискання кнопки відбувається запуск потрібних двигунів, переведення диверторів в потрібне положення, після чого починається завантаження матеріалу. Кожна кнопка відповідає за свою програму автоматичного режиму. Кнопка «Завантаження» вмикає програму для завантаження міксеру, а кнопка «Вивантаження» для вивантаження відповідно. Слід зазначити, що через використання одного компресору, неможливо одночасне виконання двох програм, згідно з джерелами [13] – [14].

Розглянемо програму автоматичного завантаження.

Функціональний блок, який відповідає за завантаження називається «DownloadingSystem».

На початку блоку представлена інструкція, яка забороняє ввімкнення комплексу, при наявних аваріях двигунів рисунок 2.39.

```
IF "AlarmExistMotors" = TRUE THEN
    "Commands".StartDownloading := FALSE;
END_IF;
```

Рисунок 2.39 – Інструкція заборони завантаження при аваріях

Як було сказано раніше, змінну на HMI панелі можно змінювати не тільки натискаючи кнопки, а також при виконанні програми. В даному випадку, якщо існують наявні помилки, оператор натискаючи кнопку «Завантаження» буде змінювати стан змінної «StartDownloading» на 1, але програму у свою чергу, побачивши наявні помилки, змінить стан на 0 та не виконає подальшу програму.

Блок «DownloadingSystem» в свою чергу поділяється на дві підпрограми, для завантаження від безтарного сховища або ручної загрузки в залежності від обраної опції на рисунку 2.36.

Якщо обрано ручне завантаження то виконується інструкція приведена на рисунку 2.40.

```
CASE "Additional".ChoiseDownload OF
    1:
        IF NOT "Commands".StartUnloading THEN
            IF "Commands".StartDownloading AND NOT "SensorMixer" THEN
                "Commands".ChangePosDiv2 := TRUE;
                IF "DivFromMan" = TRUE THEN
                    #tempDelayM10Off := FALSE;
                    IF "Commands".AutomaticM1 = TRUE THEN
                        "Commands".StartM1 := TRUE;
                    END_IF;

                    IF "Motors".Motors[0].Running AND #tempDelayM10 = FALSE THEN
                        #tempDelayM10 := TRUE;
                    END_IF;

                    IF #DelayM10.Q = TRUE THEN
                        "Commands".StartM10 := TRUE;

                    END_IF;

                END_IF;
```

Рисунок 2.40 – Інструкція програми для ручної загрузки

Як ми бачимо в залежності від обраного режиму завантаження, 1 в даному випадку означає ручне завантаження, програма обирає потрібну інструкцію. Для ручного режиму (рисунок 2.40) програма перевіряє чи не натиснута кнопка «Вивантаження», яка автоматично заборонила б завантаження.

Перший крок це приведення дивертора в напрямок завантаження міксеру. Після чого відбувається перевірка чи в правильному положенні знаходиться дивертор, якщо перевірка пройдена запускається компресор М1 та вмикається таймер, який потрібен для того щоб компресор набрав потрібний тиск для коректної роботи системи.

Після того як таймер відрахував потрібний час, вмикається дозатор М10 та комплекс починає завантаження.

Слід також детально розглянути інструкцію таймеру, яка є невід’ємною частиною будь-якої програми.

Кожен таймер це об’єкт який складається з чотирьох основних змінних рисунок 2.41.

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is set when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Рисунок 2.41 – Основні параметри таймеру в TIA Portal

1) IN – параметр старту таймеру, змінна, яка при зміненні свого стану на 1 запускає таймер, а при зміні значення на 0 перезапускає таймер;

2) PT – параметр заданого часу;

3) Q – зміна закінчення відліку, якщо значення змінної 1, це означає що таймер відрахував час, який завдається в параметрі PT;

4) ET – поточний час роботи таймеру.

Графік роботи таймеру та стан змінних приведено на рисунку 2.42.

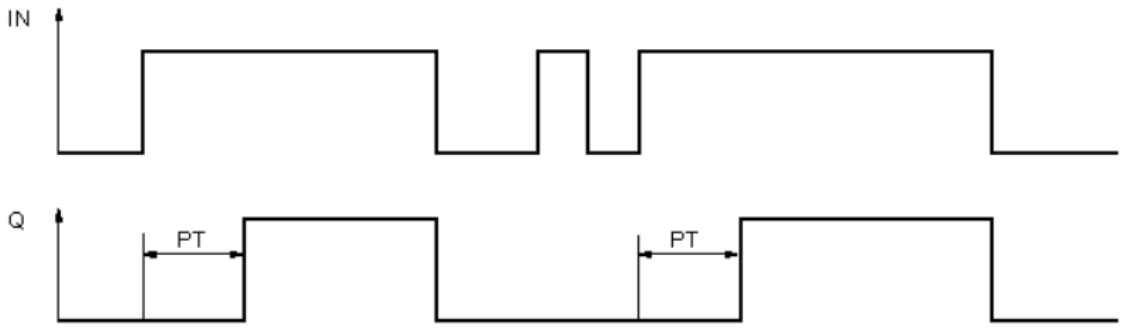


Рисунок 2.42 – Графік роботи таймеру

Як можно побачити при встановлені значення IN на 1, після відрахування часу PT параметр Q стане рівним 1, а при встановлені IN на 0, Q стане рівним 0.

Тепер більш детально розглянемо інструкцію ввімкнення M10 по таймеру на рисунку 2.43.

```
IF "Motors".Motors[0].Running AND #tempDelayM10 = FALSE THEN
    #tempDelayM10 := TRUE;
END_IF;

IF #DelayM10.Q = TRUE THEN
    "Commands".StartM10 := TRUE;

END_IF;
```

Рисунок 2.43 – Інструкція ввімкнення M10 по таймеру

Як ми бачимо, коли компресор M1 знаходиться в роботі, то змінна «tempDelayM10», яка є параметром IN, встановлює значення на 1, після чого починається відлік часу, при завершені якого змінна «DelayM10.Q» стає рівною 1 та виконує інструкцію запуску двигуна M10, джерело [15] - [16].

Після закінчення завантаження оператор натискає активну кнопку «Завантаження» ще раз, в цей момент на кнопці присутній напис «OFF» який каже яку дію виконає кнопка при натисканні. Значення змінної «StartDownloading» стає рівним 0 і починається виконання інструкції «ELSE», яка приведена на рисунку 2.44.

```
ELSE
    "Commands".StartM2 := FALSE;
    "Commands".StartM10 := FALSE;
    #tempDelayM10Off := TRUE;
    IF "Motors".Motors[10].Stop AND #DelayM10Off.Q = TRUE THEN
        IF "Commands".AutomaticM1 = TRUE THEN
            "Commands".StartM1 := FALSE;
        END_IF;


        #tempDelayM10 := FALSE;
    END_IF;
END_IF;
```

Рисунок 2.44 – Інструкція «ELSE» для ручного режиму

Ця інструкція вимикає дозатор M10 та обдув компресора M2, після чого запускає таймер, для того щоб видути залишки сировини з труб. Коли труби залишилися пустими, програма перевіряє чи зупинився дозатор M10 та після цього зупиняє компресор.

Для виконання завантаження з безтарного сховища, слід також розуміти, як взаємодіяти з вагами та отримувати поточну вагу міксеру.

Кожен модуль Siwarex при додаванні його в проект створює окрему базу даних, яка називається «WP231PR_DB» рисунок 2.45.

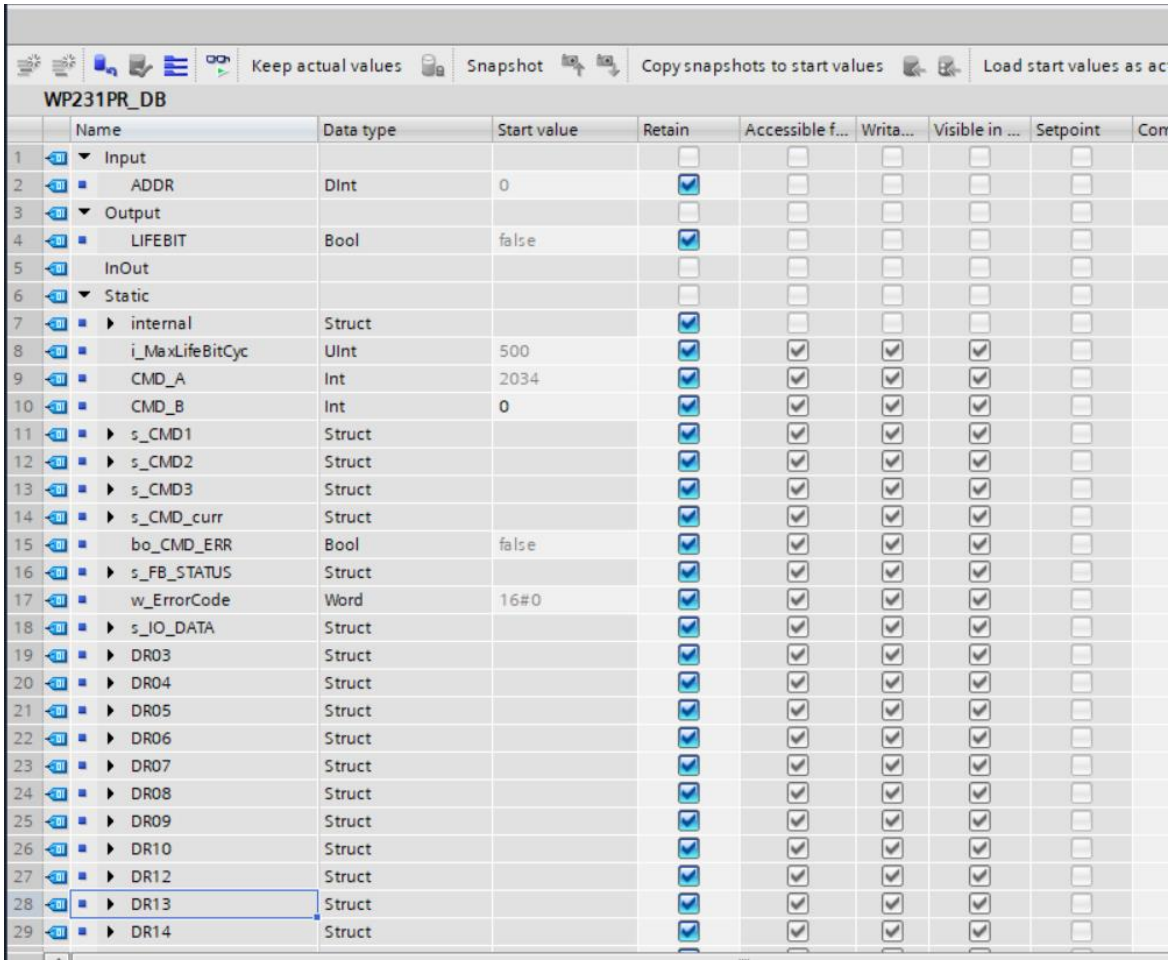


Рисунок 2.45 – База даних для вагового модуля

Ця база даних є певною програмою керування модуля. Кожна структура така як «DR03» - це налаштування модуля, які можно встановити в спеціальній програмі для взаємодії з модулем Siwarex. Модулі підтримують доволі широкий

спектр налаштувань, починаючи з чутливості датчиків, закінчуючи згладжуванням шумів.

Перед початком роботи з модулем слід звичайно налаштувати ваги, шляхом вказання початкової ваги 0 та певної калібрувальної ваги (еталону). Це робиться в спеціальний програмі налаштування ваги «Siwatool» рисунок 2.46.

Basic Parameters	
Scale name	Siwarex
Weight unit	kg
Gross indicator	B for Gross
Loading cell type	Strain gauge analogue
Restriction code	none
Minimum weight (in d)	20
Maximum weight	100.0
Resolution d	0.1
Scale characteristic curve	
Calibration weight 0	0.0
Calibration weight 1	100.0
Calibration weight 2	0.0
Calibration digits 0 (measured)	0
Calibration digits 1 (measured)	2000000
Calibration digits 2 (measured)	0

Рисунок 2.46 – Програмне забезпечення «Siwatool»

На рисунку 2.46 представлений розділ з базовими параметрами. Базові параметри включають в себе налаштування вагових одиниць, тип тензодатчиків, роздільна здатність, максимальна вага, тощо. Важливим параметром є вага калібрування, яка має бути еталонною для отримання точних вимірювань.

У процесі калібрування задається перша точка нулю, тобто точка при якій ми хочемо бачити на вагах нуль з урахуванням конструкцій, що вже стоїть на вагах. Після встановлення першої точки, встановлюється еталонна вага та вказується друга точка.

Після всіх маніпуляцій будується спеціальна пряма, за якою в подальшому буде розраховуватися вага в залежності від сили сигналу рисунок 2.47.

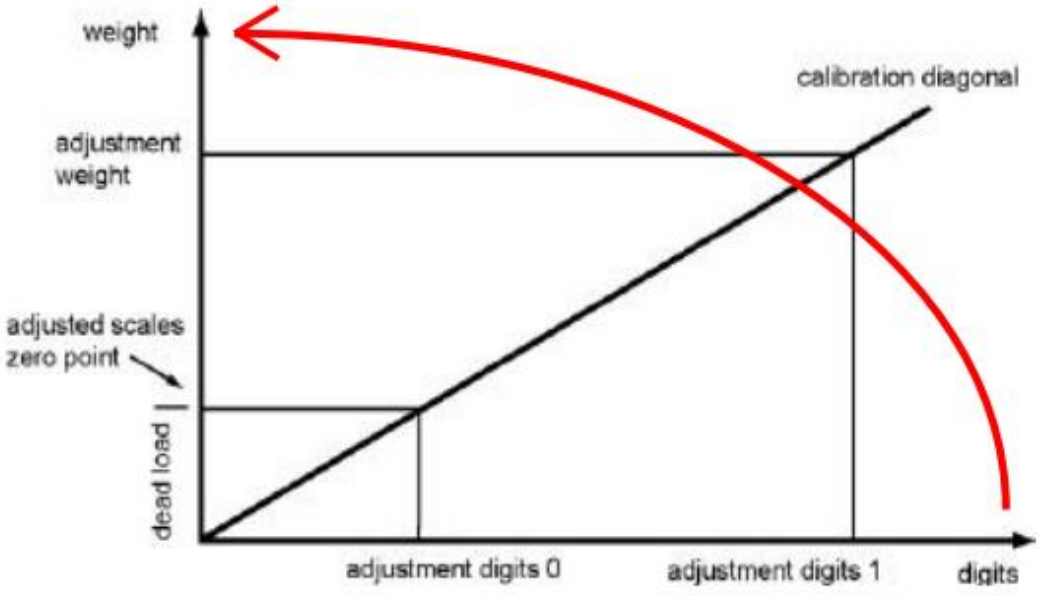


Рисунок 2.47 – Калібрувальна пряма залежності ваги від сигналу.

Після налаштування ваги, можно звернутися до бази даних, яка була створена модулем «SIWAREX» та в полі «PROCESS_VAL_1» побачити поточну вагу рисунок 2.48.

18	s_IO_DATA	Struct	
19	COORDINATION	Byte	16#0
20	APPL_ID_ACTUAL	Byte	101
21	ERROR_CODE	UInt	0
22	SCALE_STATUS_1	Struct	
23	SCALE_STATUS_2	Struct	
24	PROCESS_VAL_1	Real	0.0

Рисунок 2.48 – Поточна вага, яку показує модуль

Поточна вага також відображається на HMI панелі рисунок 2.49.

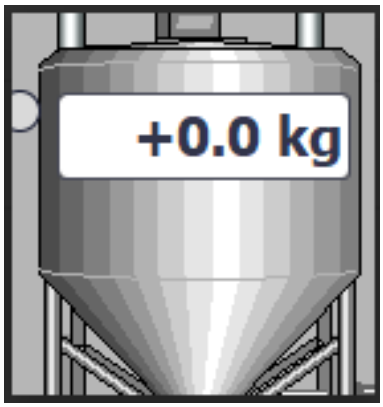


Рисунок 2.49 – Поточна вага міксеру

У випадку коли обране завантаження з безтарного сховища, перш за все треба обрати рецепт завантаження на екрані рецепту. На цьому екрані слід обрати загальну вагу сировини, яка відвантажується з ємностей та відсоткове співвідношення для будь-якої ємності. Програма автоматично розрахує вагу сировини яку слід завантажувати рисунок 2.50.

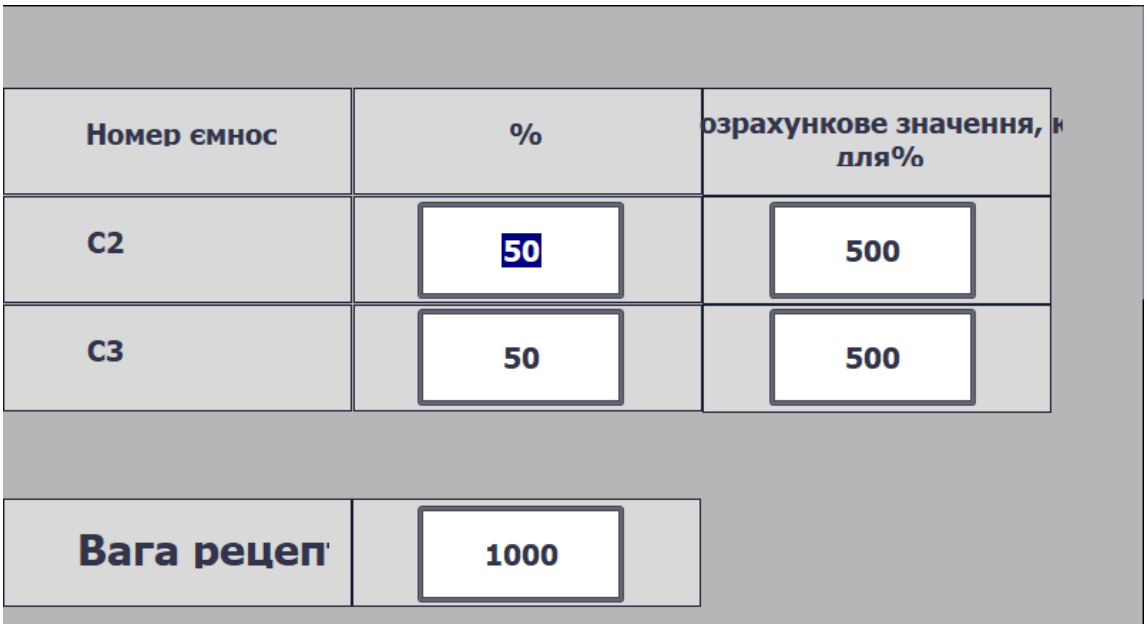


Рисунок 2.50 – Екран обрання рецепту для завантаження

Після того як обрано рецепт та натиснута кнопка «Завантаження», починається виконання інструкції на рисунку 2.51.

```
        IF "RecipesDB".WeightRecipe <> 0 THEN
            "Commands".StartB2 := TRUE;
        END_IF;

        IF "RecipesPercent".WeightPercentC2 <> 0 THEN

            IF "WorkB2" THEN

                "Commands".StartShP2 := TRUE;
                IF "WorkShP2" = TRUE THEN
                    #tempTonSP := TRUE;
                END_IF;


            END_IF;
            IF #TonSP.Q THEN
                #tempTonSP := FALSE;
                "Commands".ChangePosSh2 := TRUE;
            END_IF;
            IF "Weight".weight >= "CalculatedWeight".WeightC2 THEN
                "Commands".StartShP2 := FALSE;
                "Commands".ChangePosSh2 := FALSE;
                #tempBlowDown := true;

            END_IF;

            IF #TonBlowDown.Q = TRUE THEN
                #tempBlowDown := FALSE;
                #tempTonB2Off := TRUE;
                IF "RecipesPercent".WeightPercentC3 <> 0 THEN
                    "Services".ModeRecipe := 2;
                ELSE
                    "Services".ModeRecipe := 3;
                END_IF;


            END_IF;
```

Рисунок 2.51 – Інструкція завантаження з безтарного сховища

На початку вмикається компресор, після того як компресор набрав потрібні оберти, відкривається заслінка та вмикається дозатор, після чого починається завантаження міксеру. Коли вага міксеру буде більше ніж вага задана за рецептом, завантаження закінчиться рисунок 2.52.

```
IF "Weight".weight >= "CalculatedWeight".WeightC2 THEN
    "Commands".StartShP2 := FALSE;
    "Commands".ChangePosSh2 := FALSE;
    #tempBlowDown := true;

END_IF;
```

Рисунок 2.52 – Інструкція закінчення завантаження

Аналогічно відбувається завантаження для іншого силосу. Коли вага міксеру буде дорівнювати вазі рецепту, кнопка «Завантаження» буде виключена,

тим самим завантаження закінчиться.

Другий основний режим комплексу є «Вивантаження».

При натисканні кнопки «Вивантаження» виконується інструкція на рисунку 2.53.

```
IF "Commands".StartUploadToLine = TRUE THEN
    IF "Commands".UploadTanks = 0 THEN // Statement section IF
        IF "DownloadRequest" = FALSE AND "SwichM9" = true AND NOT "SensorShNTank2" = FALSE THEN
            IF #tempReq=TRUE THEN
                "Commands".StartM9 := true;
            END_IF;
            "Commands".ChangePosTank2 := TRUE;
            "klAirTank2" := TRUE;
        ELSE
            "Commands".StartM9 := FALSE;
            IF #tempReqOff=FALSE THEN
                "Commands".ChangePosTank2 := FALSE;
            END_IF;
            "klAirTank2" := FALSE;
        END_IF;

    END_IF;
    IF "Commands".UploadTanks = 1 THEN // Statement section IF
        IF "DownloadRequest"=FALSE AND  "SwichM8" = TRUE AND NOT "SensorShNTank1" = FALSE THEN
            IF #tempReq = TRUE THEN
                "Commands".StartM8 := true;
            END_IF;
            "Commands".ChangePosTank1 := TRUE;
            "klAirTank1" := TRUE;
        ELSE
            "Commands".StartM8 := FALSE;
            IF #tempReqOff=FALSE THEN
                "Commands".ChangePosTank1 := FALSE;
            END_IF;

            "klAirTank1" := FALSE;
        END_IF;
```

Рисунок 2.53 – Інструкція для режиму вивантаження

Інструкція перевіряє чи натиснута кнопка «Вивантаження» та в залежності від того, яка розхідна ємність обрана вмикає розвантажувальний шнек та відкриває заслінку відповідної ємності. Все вивантаження відбувається згідно зовнішньому запиту «"DownloadRequest"», якщо зовнішній запит має логічний нуль, то відбувається вивантаження до того моменту поки кнопка «Вивантаження» не буде віджата.

Отже основна частина роботи комплексу відводиться саме автоматичній програмі, учать в якій людина приймає тільки на етапі налаштування та вибору режиму роботу комплексу.

В момент виконання програми, поточний стан датчиків, наповнення,

аварії, цілком контролюються програмою та відображаються на HMI панелі, що забезпечує неперервну роботу комплексу та зручно взаємодію з ним.

2.5 Обробка та відображення помилок

Виникнення помилок – невід'ємна частина будь-якої системи, отже слід розробити правильну систему для їх обробки, відображення та реакції на них.

Всі помилки, які можуть виникнути під час роботи комплексу, інтегровані в програму та будуть відображені на HMI панелі.

Для відображення помилок використовується спеціальний вбудований екран рисунок 2.54.

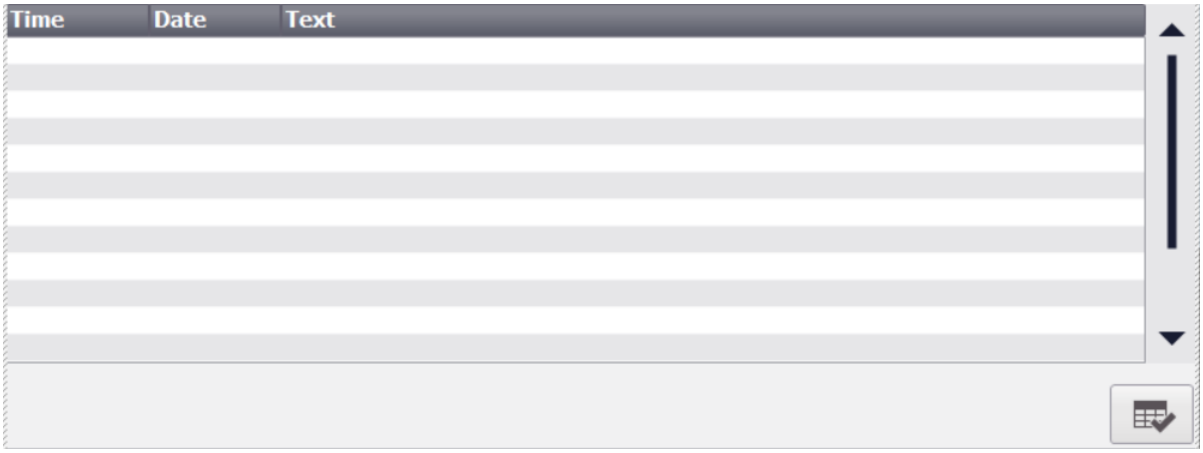


Рисунок 2.54 – Екран відображення помилок

На цьому екрані будуть відображатися наявні помилки з відповідними мітками часу, дати та тексту помилки.

Для початку слід створити зміну типу «Word». Зміна цього типу займає 16 біт в пам'яті і в свою чергу може тримати інформацію про 16 помилок одночасно, отже, якщо в системі наявні більш ніж 16 помилок слід створювати нову зміну для зберігання їх.

Розглянемо процес створення помилки для кнопки аварійної зупинки.

В загальному будь-які безпекові атрибути встановлюються в розрив електричного кола, це робиться з тією метою щоб при пошкоджені кола, система

вийшла з ладу так само як і при натисканні стопової кнопки.

Отже, помилка яка викликається натискання стопової кнопки буде виникати, якщо на відповідному дискретному вході контролера буде логічний нуль. На рисунку 2.55 показано присвоєння відповідному біту змінної «Alarm» значення логічної одиниці, якщо значення входу стопової кнопки буде ріне нулю.

```
34
35  "Alarm".%X15 := NOT "EmergencyStopBtn";// кнопка аварійної зупинки
36
```

Рисунок 2.55 – Присвоєння значення помилки стопової кнопки

Після чого, відповідний біт можно використовувати в вікні помилок, вказавши йому текст помилки, її клас та біт змінною, з якої ця помилка буде з’являтися рисунок 2.56.

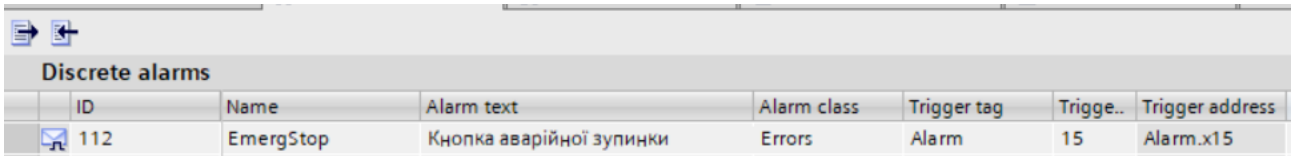


Рисунок 2.56 – Створення помилки в HMI панелі

Кожна посилка має наступні основні параметри:

– «ID» – унікальний номер помилка;
– «Name» – назва помилки;
– «Alarm text» – текст помилки;
– «Alarm class» – клас помилки;
– «Trigger tag» – зміна, яка зберігає інформацію про помилки;
– «Trigger bit» – біт, який зберігає інформацію про конкретну помилку;
– «Trigger address» – абсолютна адреса зберігання помилки.

Після реєстрації помилки, при виникненні логічного нуля на вході, який

відповідає за стан стопової кнопки, на екрані помилок з’явиться відповідна помилка рисунок 2.57.

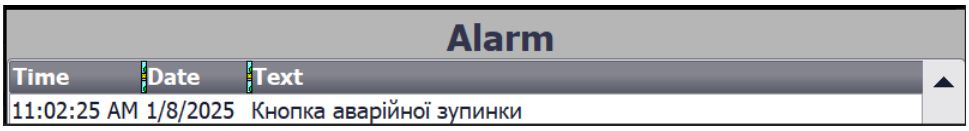


Рисунок 2.57 – Поява помилки про натискання стопової кнопки

Після усунення помилка, а саме вимкненню стопової кнопки, помилка автоматично пропаде.

Важливим моментом обробки помилок є інтегрування їх в загальну систему роботи комплексу, адже помилки є невід’ємною частиною роботи і не завжди означають некоректну роботу комплексу, як зазначається у джерелі [17].

Як було згадано раніше, в міксері присутній датчик верхнього рівня, який служить індикатором заповнення міксеру до максимального рівня, отже при його спрацюванні виконується інструкція на рисунку 2.58 та забороняється завантаження.

```
66
67 IF "SensorMixer" THEN
68
69     "Commands".StartDownloading := FALSE;
70
71 END_IF;
```

Рисунок 2.58 – Інструкція заборони завантаження при спрацьовуванні датчика рівня

В свою чергу при спрацьовуванні датчику рівня на екрані відображається спеціальний індикатор, який це сигналізує рисунок 2.59.

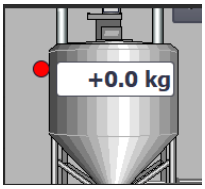


Рисунок 2.59 – Індикатор спрацювання датчика рівня в міксері

2.6 Розробка регулятора для двигуна

Уявимо ситуацію, оператор виставляє значення частоти шнеку для подачі сировини в розхідні ємності. Система працює нормально так як подача збалансована з розходом, але в подальшому може виникнути ситуація коли розхід змінюється і з системи починають відбирати більше сировини ніж вона дозволяє. Для вирішення цієї проблеми знадобиться розробити регулятор для двигуна, зворотнім зв’язком для якого буде ультра-звуковий датчик.

Схема керування приводом зі зворотнім зв’язком на рисунку 2.60.

Рисунок 2.60 – Схема керування двигуном

Основним зворотнім зв’язком виступаю сигнал з датчика рівня, який приймає значення від 0 – 100%, що означає що якщо заданий рівень почав знижуватися розхід системи збільшився, а отже треба підняти частоту двигуна для відновлення балансу систему.

Для модуляції регулятора зробимо схему в середовищі Simulink, яка показана на рисунку 2.61.

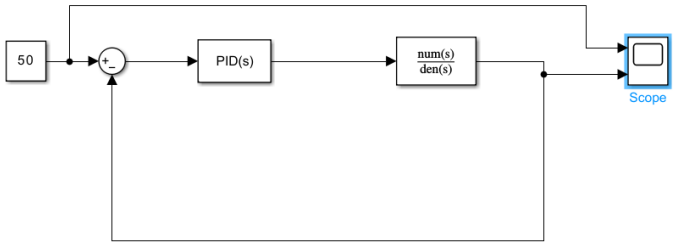


Рисунок 2.61 – Схема ПІД-регулятора

Розробимо керування двигуном за допомогою пропорційного елементу регулятора рисунок 2.62.

```
#K1 := (#LevelTask - "LevelTanks".LevelTank1) * 0.5;

#FriqOut := #FriqUst + #K1;
```

Рисунок 2.62 – Пропорційний елемент регулятора

В даному випадку ми беремо поточну помилку між заданим рівнем в ємності та реальним, після чого помножуємо його на коефіцієнт пропорційності та додаємо результат до нашої установки частоти. Результат такого регулятора можно побачити на рисунку 2.63.

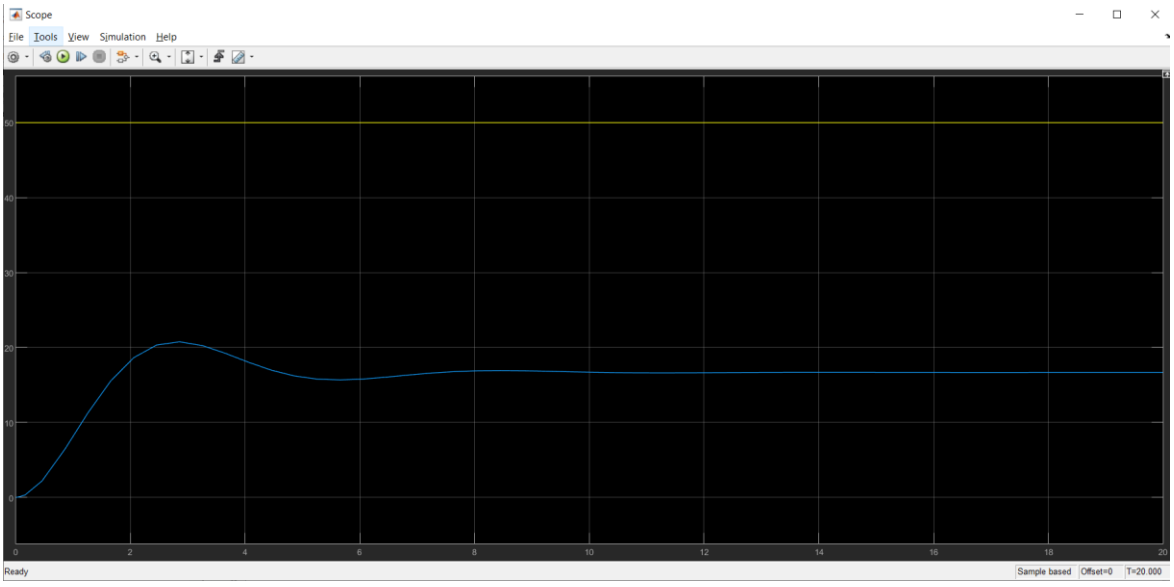


Рисунок 2.63 – Пропорційний регулятор двигуна

Як можно побачити, при нульовому диференційному та інтегральному коефіцієнті, система не підтримую задану величину, а при збільшені пропорційного коефіцієнту, двигун починає занадто часто змінювати свою частоту, що призводить до зносу обладнання.

Для вирішення цієї проблеми розробимо диференційну частину, яка представлена на рисунку 2.64.

```
#Error := #LevelTask - "LevelTanks".LevelTank1;
#K2 := (#Error - #PrevError) * 0.7;
#PrevError := #Error;
```

Рисунок 2.64 – Диференційний регулятор

В даному випадку визначається помилка між поточним значенням та попереднім, яка потім помножується на диференційний коефіцієнт, а нова помилка записується в попередню, після чого буде зрівнюватися з наступною помилкою.

Результат перехідного процесу з додаванням диференційною частини на рисунку 2.65.

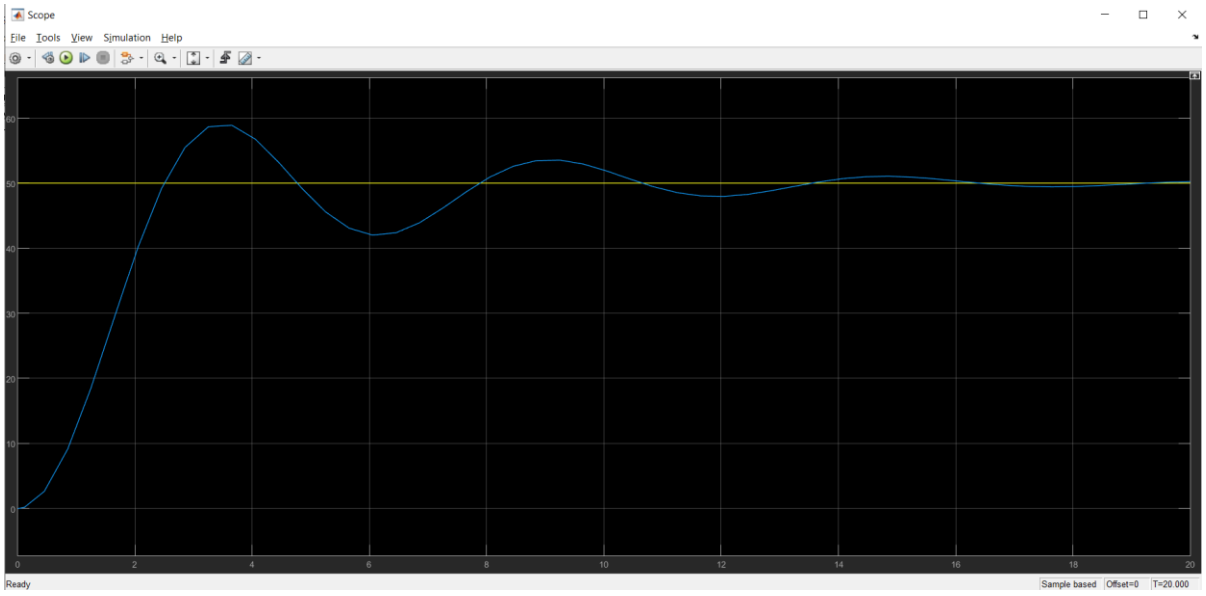


Рисунок 2.65 – Пропорційно-диференційний регулятор

Як можно побачити, в такому випадку ми досягли заданої величини але в свою чергу регулятор виходить за її межі та витрачає приблизно п’ятнадцять секунд на стабілізацію, що не відповідає нашим вимогам.

Фінальна частиною стане додавання інтегральної частини до нашого регулятора на рисунку 2.66.

```
#K3 := #K3 + (#LevelTask - "LevelTanks".LevelTank1) * 0.2;
```

Рисунок 2.66 – Інтегральний регулятор

Інтегральний регулятор носить в собі характер накопичення помилок, отже має найнижчий коефіцієнт перетворення.

Результат роботи регулятора можно побачити на рисунку 2.67.

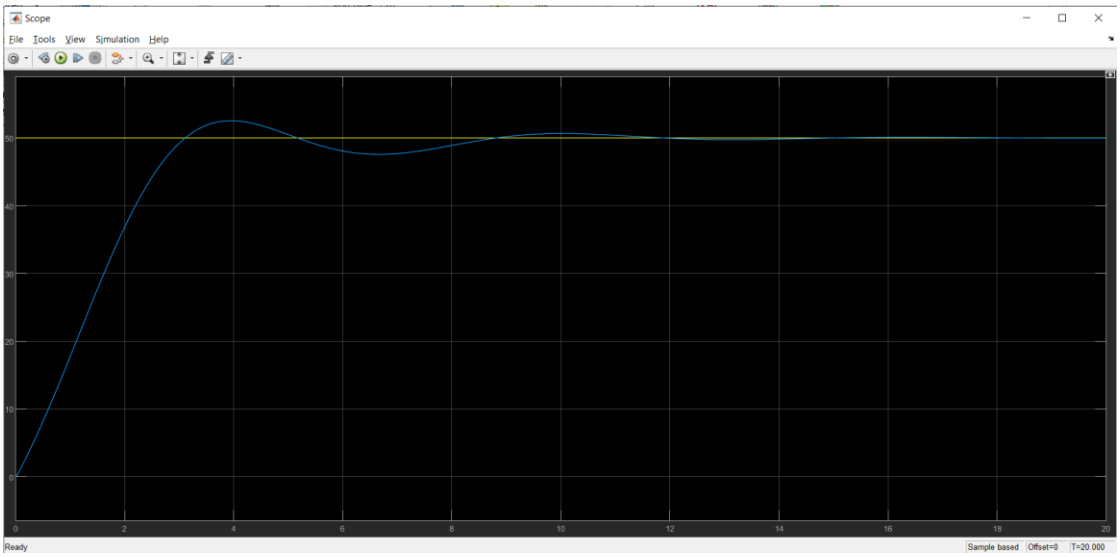


Рисунок 2.67 – Пропорційно-інтегрально-диференційний реуглятор

В даному випадку ми отримали час стабілізації системи вісім секунд, що дозволяє вчасно реагувати на зміни розходу сировини, що в свою чергу забезпечує стабільну роботу установи. Розроблений регулятор також забезпечує плавність перехідного процесу, що в свою чергу запобігає різким змінам частот двигуна, що забезпечує його довготривалу роботу.

2.7 Висновки до другого розділу

У другому розділі роботи було проведено ознайомлення з програмним забезпеченням для програмування обраного контролеру. Розробили архітектуру проекту. Розробили системи для взаємодії с аналоговими сигналами, розглянули принцип роботи таймерів та принцип налаштування тензодатчиків.

Розробили ПЗ для керування моторів та моніторингу їх поточного стану, а також розробили регулятор для моторів, які це потребують.

Розробили ПЗ для керування комплексом в автоматичному режимі, яке в свою чергу має гнучкі налаштування та не вимагає участі людини після включення програми. Програма в свою чергу є досить гнучкою та легко змінюється під потреби.

Намалювали та прив’язали до ПЗ контролеру людино-машинний інтерфейс, який слугує зв’язком між оператором та комплексом. Інтерфейс включає в себе основні параметри та налаштування комплексу, ручний режим та відображає його поточний стан. Цей інтерфейс зрозумілий без великих витрат часу на його вивчення, що є величезною перевагою, яка дозволяє не витрачати багато часу на навчання оператора.

Окремим розділом розробили блок, який відповідає за відображення помилок комплексу та інтегрували ці помилки в роботу комплексу. Розробили індикатори для відображення аварійних датчиків комплексу.

Розробили ПІД-регулятор для регулювання процесу наповнення розхідних ємностей. Підібрали коефіцієнти для плавного регулювання перехідного процесу, що забезпечує швидку реакцію систему на відхилення від поточного розходу та плавного регулювання його.

У цьому розділі були виконані всі поставлені задачі для ПЗ комплексу, яке в свою чергу на практиці показало себе, як надійне та ефективне ПЗ для промислових цілей.

3 РОЗРАХУНОК НАДІЙНОСТІ ТА БЕЗВІДМОВНОСТІ КОМПЛЕКСУ

3.1 Загальна характеристика безвідмовної роботи та збір даних

Безвідмовна роботи системи – важливий атрибут будь-якої автоматизованої системи. Безперервна робота системи впливає на всі аспекти виробництва, в тому числі на безпеку, якість продукції, економіку підприємства.

Всі вироби мають таку характеристику як «інтенсивність відмов», яка показує наскільки часто обладнання буде видавати помилку під час роботи тим самим виводячи систему з ладу.

Окрім «інтенсивності відмов» також присутня характеристика «інтенсивність відновлення», яка показує скільки часу потрібно для виправлення відмови того чи іншого приладу.

Саме ці два параметри і дають загальну оцінку надійності системи в цілому та дозволяють розрахувати коефіцієнти готовності та вірогідність безвідмовної роботи.

В загальному існують декілька методів аналізу надійності системи, а саме:

- метод логічних діаграм;
- аналіз дерева відмов;
- Марковський аналіз.

Початок кожного методу однаковий та являє собою збір даних про інтенсивність відмов та відновлення. Після чого слід розрахувати так звані MTBF та MTBR, тобто середній час напрацювання на відмову, середній час відновлення, згідно з джерелом[18].

Далі ми будемо використовувати метод аналізу логічних діаграм. Цей метод дозволяє оцінити потрібний рівень надійності

Отже розрахунок та прогнозування безвідмовної роботи та надійності системи є невід’ємним атрибутом при розробленні ПЗ.

Побудуємо логічну блок-схему для нашої системи, яка приведена на рисунку 3.1.

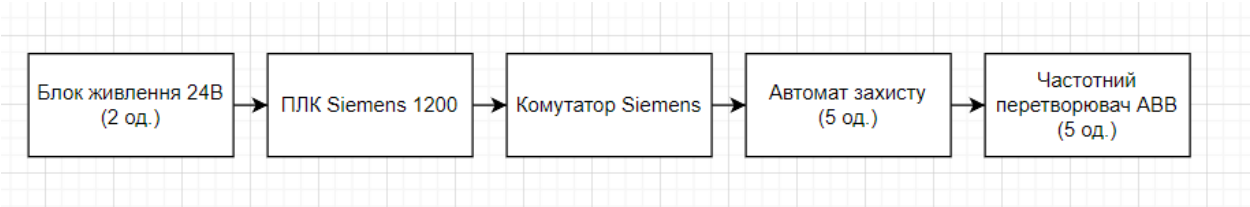


Рисунок 3.1 – Логічна блок-схема системи керування

Зі схеми маємо послідовний тип системи, що означає що відмова в будь-якому етапі, призводить до відмови всієї системи.

Основними даними для розрахунку надійності системи є середній час до першої відмови та час до усунення відмови. Ці дані отримуються експериментальним шляхом в ході роботи системи та приведені в таблиці 3.1.

Таблиця 3.1 – час до першої відмови елементів системи

Обладнання	Середній час першої відмови, $T_в$, год	Час відновлення після відмови, $T_{від}$, год
ПЛК Siemens 1200	8000	1
Блок живлення	5000	2
Комутатор Siemens	2700	0.5
Автомат захисту	700	0.2
Частотний перетворювач ABB	800	0.2

3.2 Розрахунок надійності та вірогідності безвідмовної роботи

Після отримання цих даним, водиться три основних характеристики для оцінки надійності.

Перше це інтенсивність відмов, яка розраховується за формулою:

$$\lambda = \frac{1}{T_в}, \tag{3.1}$$

де λ – інтенсивність відмов;

$T_в$ – час до першої відмови.

Друге це інтенсивність відновлення, формула:

$$\mu = \frac{1}{T_{від}}, \tag{3.2}$$

де μ – інтенсивність відновлення;

$T_{від}$ – час відновлення.

Трете це коефіцієнт готовності, формула:

$$К = \frac{\mu}{\mu+\lambda}, \tag{3.3}$$

де К – коефіцієнт готовності;

λ – інтенсивність відмов;

μ – інтенсивність відновлення.

У випадку повторення однакових елементів, їх сумарна інтенсивність розраховується за формулою:

$$\lambda_с = \sum \lambda_i \tag{3.4}$$

Коефіцієнт готовності в такому випадку розраховується зо формулою:

$$К_с = \prod К_i \quad (3.5)$$

Користуючись формулами (3.1), (3.2), (3.3), (3.4), (3.5), розрахуємо відповідні дані та занесемо результати у таблицю 3.2.

Таблиця 3.2 – результат розрахунків інтенсивності відмов, відновлення та коефіцієнту готовності

Обладнання	Інтенсивність		Коефіцієнт готовності
	Відмов, $\lambda \cdot 10^{-4}$	Відновлення, μ	
ПЛК Siemens 1200	1.25	1	0.9998
Частотний перетворювач ABB	10	0.5	0.998
Блок живлення	7.4	2	0.9996
Комутатор Siemens	14.2	5	0.9997
Автомат теплового захисту	62.5	5	0.9987

Отримавши ці результати можемо розрахувати інтенсивність відмов всієї системи за формулою (3.4).

$$\lambda_з = 1.25 * 10^{-4} + 10 * 10^{-4} + 7.4 * 10^{-4} + 14.2 * 10^{-4} + 62.5 * 10^{-4} = 95.3 * 10^{-4}$$

Загальний коефіцієнт готовності розрахуємо за формулою (3.5).

$$К_з = 0.9998 * 0.998 * 0.9996 * 0.9997 * 0.9987 = 0.9958$$

Після отримання цих даних, може провести аналіз за допомогою міжнародного стандарту IEC 61508, який використовується для оцінки промислових систем, що описується в джерелі [19] – [20].

Згідно цього стандарту, існують чотири рівня надійності системи, які називають рівні інтегральної надійності SIL, в залежності від вірогідності відмови системи. Стандарт має наступні рівні:

- захист обладнання та продукції;
- захист обладнання, продукції та захист від травматизму;
- захист персоналу та населення;
- захист від загальної катастрофи.

Згідно таблиці 3.3, в якій приведені рівні SIL та відповідні коефіцієнти для отримання системою того чи іншого рівня.

Таблиця 3.3 – Опис вимог та характеристик рівнів SIL

SIL	Вірогідність небезпечної відмови	Потрібний коефіцієнт готовності	Частота небезпечних відмов, λ , 1/год	Фактор зниження ризику, T , 1/ λ
1	$10^{-2} – 10^{-1}$	90% - 99%	$10^{-6} – 10^{-5}$	від 10 років до 100 років
2	$10^{-3} – 10^{-2}$	99% - 99.9%	$10^{-7} – 10^{-6}$	від 100 років до 1000 років
3	$10^{-4} – 10^{-3}$	99.9% - 99.99%	$10^{-8} – 10^{-7}$	від 1000 років до 10000 років
4	менш ніж 10^{-4}	більш ніж 99.99%	менш ніж 10^{-8}	більш ніж 10000 років

Згідно таблиці 3.3 можемо зробити висновок, що при отриманому коефіцієнті готовності у 99.5%, система відноситься до другого рівня SIL.

Прорахувавши інтенсивність відмови системи, можемо розрахувати вірогідність безвідмовної роботи системи:

$$P(t) = e^{-\sum \lambda_i} \tag{3.6}$$

Вірогідність безвідмовної роботи системи з n послідовних елементів дорівнює експоненті в від'ємній степені суми всіх інтенсивностей відмов кожного окремого елементу. В нашому випадку за формулою (3.6).

$$P(t) = e^{-95.3*10^{-4}} = 0.9905$$

Отже наша система має досить високу вірогідність безвідмовною роботи у 99.5%.

3.3 Висновки до третього розділу

Будь-яка система, особливо промислова, вимагає певний ступінь надійності, адже будь-яка поломка чи затримка системи призводить до економічних збитків.

На першому етапі цього розділу, експериментальним шляхом отримали частоту відмов основних елементів комплексу, які відмовляли впродовж його експлуатації (півтора року) та побудували логічну блок-схему керування комплексом.

Зібравши всі необхідні дані провели розрахунки вірогідності безвідмовної роботи системи, що склала 99.5%.

Провівши ці розрахунки, користуючись рівнями SIL, зробили висновок, що система відноситься до другого рівня, що є високим показником для промислових систем.

4 ОХОРОНА ПРАЦІ

4.1 Техніка безпеки при роботі з комплексом

Комплекс безтарного сховища є системою з багатою кількістю механізмів та електроніки, які можуть нести певні ризики для здоров’я людини.

Кожен оператор та технічний співробітник має дотримуватися інструкцій для безпечною роботи з комплексом.

Перед початком роботи кожен працівник має пройти відповідний інструктаж та отримати відповідні інструкції для роботи.

При взаємодії з комплексом під час роботи слід дотримуватися певних правил для запобіганню шкоди для здоров’я.

Найголовнішим елементом безпеки є стопова кнопка рисунок 4.1.

Рисунок 4.1 – Стопова кнопка

При натисканні стопової кнопки, всі механізми примусово зупиняються незважаючи на те, в якому стані знаходиться програма комплексу.

Окрім стопової кнопки, яка зупиняє весь комплекс, також встановлені спеціальні ручні перемикачі для від’єднання живлення з деяких механізмів, для запобігання їх ввімкненню за допомогою програми рисунок 4.2.

Рисунок 4.2 – Ручний вимикач для шнеку вивантаження

Такі вимикачі потрібні для безпечного усунення збоїв комплексу, без зупинки всієї системи. При забиванні шнеку, оператор може зняти живлення з нього для безпечної очистки його, після чого комплекс поновить роботу в штатному режимі, згідно джерела [21].

Якщо виникає потреба виправити несправність, яка вимагає виключення механізму шляхом зняття живлення в електричній шафі, слід викликати відповідного працівника, який має допуск для проведення електричних робіт. Після того як працівник від’єднає живлення з механізму, можно переходити до відповідних маніпуляцій, таких як прокручення мотору механічно.

При роботі з комплексом в режимі обслуговування чи модернізації слід виконати відповідні підготовчі дії.

По-перше при роботі з комплексом треба виключити живлення головним

рубильником в електричній шафі.

Це запобігає випадковим включенням механізмів та як наслідки отриманню виробничих травм, що вказано в джерелі[22].

По-друге слід перекрити повітря та заблокувати всі рухомі механізми для запобігання небажаного спрацьовування пневматики.

При роботі з комплексом слід встановити відповідні таблички, які сигналізують, що не можна вмикати живлення, тому що в даний момент проводяться певні роботи, такі таблички представлені на рисунку 4.3.

Рисунок 4.3 – Табличка про заборону включення обладнання

При роботі з електронікою слід дотримуватися всіх правил, які були названі вище та дотримуватися відповідних правил роботи на електричному обладнані.

Перш за все, людина, яка працює з електричним обладнанням, має мати відповідну групу дозволу та підписати відповідні робочі інструкції.

Забороняється вимикати чи «коротити» небажані системи безпеки, так як це може призвести до небажаних наслідків та травм.

Забороняється також змінювати налаштування частотних перетворювачів, номінальних струмів автоматів захисту двигунів під час роботи комплексу, так

як це може призвести до його зупинки.

При роботі з програмою спеціалістами АСУ ТП слід пам’ятати, що завантаження програми в деяких випадках потребує переведення контролеру в режим зупинки, що призводить до зміни всіх його виходів на логічні нулі, що в свою чергу призводить до зупинки комплексу.

Отже при перезавантаженні програми слід переконатися що:

– на момент перезавантаження програми комплекс знаходиться в режимі простою та не виконує свої функції;

– під час оновлення ПО над комплексом не проводяться технічні роботи;

– перекриті всі пневматичні механізми, а двигуни вимкненні автоматами захисту, для запобігання небажаного спрацьовування.

Після оновлення ПО, слід переконатися що всі захисні функції, а саме кнопки аварійної зупинки, кінцеві вимикачі, механічні перемикачі, працюють нормально та виконують задані функції

Дотримання всіх правил забезпечує безпеку працівників та запобігання травм та нещасних випадків та є обов’язковим для будь-якого підприємства та установи. Окрім цього дотримання технік безпеки дозволяє подовжити життя обладнанню та комплексу в цілому, що впливає на економіку підприємства та якість технологічних процесів.

4.2 Висновки до четвертого розділу

Спираючись на півтора річний досвід роботи з комплексом та загально прийнятими правилами роботи на промислових об’єктах розробили певні правила роботи с системою.

По-перше описали основні безпекові елементи системи, такі як: стопові кнопки, механічні вимикачі, які потрібні для екстрених ситуацій.

По-друге розробили певні правила роботи з комплексом для оператора, електрика та інженера з автоматизації.

Описали основні ризики при роботі з комплексом та шляхи їх мінімізації при дотриманні відповідних інструкцій та правил, які окрім збереження людського здоров’я, збережуть обладнання та компоненти комплексу.

ВИСНОВКИ

Під час виконання магістерської роботи, було проаналізовано систему безтарного сховища сировини на предмет актуальності та економічної обґрунтованості, розглянуто основні цілі комплексу, створено ПЗ для роботи з комплексом та проведено аналіз надійності та безвідмовності комплексу.

На першому етапі було розглянуто конструкцію комплексу безтарного сховища. Розглянуті основні елементи та структури комплексу, такі як силоси, дивертори. Розглянуто системи механічних комунікацій та виконуючих механізмів.

На другому етапі розглянута компонента база комплексу. Розглянуті датчики та електроніка, яка використовується в комплексі, розглянуті основні види передачі аналогових сигналів та взаємодії з ними на рівні програми. Представлено та обґрунтовано вибір основного контролеру комплексу та HMI панелі, за допомогою якої відбувається взаємодія оператора з комплексом.

Після перших двох етапів, можно зробити висновки, що комплекс є дуже варіативним та гнучким у виборі компонентної бази, як і механічної так і електронної, та може адаптуватися під потреби будь-якого замовник.

У роботі було представлено процес розробки програмного забезпечення для роботи з комплексом. Спочатку була проведена конфігурація обладнання та розподілення основних логічних входів-виходів ПЛК, після чого було розроблено програму для керування двигуном, як окремим одиничним механізмом. Розроблено також основний функціонал для обробки аналогових сигналів та взаємодії з ними.

Другим етапом розробки ПЗ стало створення програми для автоматичної роботи комплексу, яка включає в себе два основних етапи роботи та є повністю автоматичною за винятком початкових налаштувань.

Намальований людино-машинний інтерфейс для взаємодії оператора з комплексом, який у свою чергу відображає поточний стан комплексу, режим

роботи, наявні помилки. В інтерфейсі також присутні основні налаштування комплексу та можливість його роботи в ручному режимі.

Розроблене програмне забезпечення виявилося доволі гнучким та легко піддається змінам. Програмне забезпечення показало чудові результати на довгому відрізку часу, працювавши належним чином та без помилок зі сторони розробника. Людино-машинний інтерфейс інтуїтивно зрозумілим, що дозволяє скоротити час навчання персоналу та забезпечити якісне функціювання комплексу.

Проведено аналіз надійності роботи комплексу. Зібравши дані про інтенсивності відмов та відновлення основних елементів комплексу було прораховано коефіцієнт готовності та вірогідність безвідмовної роботи комплексу. Результати розрахунків показують, що комплекс відноситься до другого рівня SIL та є дуже надійною системою для своїх задач.

Отже розроблений комплекс не тільки виконую всі поставлені задачі, такі як зниження складських площ та зв’язок між двома виробничими лініями, що дозволяє значно покращити економічний стан установ, які використовують комплекс, а також є дуже варіативною системою, яка легко може включити в себе елементи під кожну специфічну задачу. Комплекс виявився доволі надійним, що дозволяє майже виключити проблеми с зупинками, які несуть за собою економічні втрати. Завдяки правильному підбору обладнання та розробленому ПЗ комплекс піддається швидкою модернізації за потреби.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке силос? [Електронний ресурс] /– Режим доступу: www / URL: https://krupaik.com.ua/articles/chto-takoe-silos.

2. Технологія торгівлі та послуг : навч. посіб. / В. В. Лісіца, О. М. Михайленко, Ю.В. Іванов. – Полтава : ПУЕТ, 2008. – 175 с.

3. Про вищу освіту [Електронний ресурс] : Закон України від 01.07.2014 р. № 1556-VII // Верховна Рада України : офіційний веб-портал. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1556-18#Text. – Станом на 18.09.2021. – Назва з екрану.

4. Положення про організацію освітнього процесу в ХНУРЕ [Електронний ресурс] : наказ ректора ХНУРЕ від 27.11.2020 р. № 400 // Нормативно-правова база ХНУРЕ : офіційний веб-портал. – Режим доступу: https://nure.ua/universytet/normativno-pravova-baza#id13. – Станом на 18.09.2021. – Назва з екрану.

5. Положення про протидію академічному плагіату в Харківському національному університеті радіоелектроніки [Електронний ресурс] : наказ ректора ХНУРЕ від 28.04.2017 р. № 290 // Нормативно-правова база ХНУРЕ : офіційний веб-портал. – Режим доступу: https://nure.ua/universytet/normativnopravova-baza#id13. – Станом на 18.09.2021. – Назва з екрану.

6. Положення про порядок створення та організацію роботи екзаменаційних комісій з атестації здобувачів вищої освіти ступенів бакалавр, магістр (спеціаліст) [Електронний ресурс] : наказ ректора ХНУРЕ від 09.02.2015 р. № 40 // Нормативно-правова база ХНУРЕ : офіційний веб-портал. – Режим доступу: https://nure.ua/universytet/normativno-pravova-baza#id13. – Станом на 18.09.2021. – Назва з екрану.

7. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – [На заміну ДСТУ 3008-95 ; чинний від 2017-07-01]. – К. : ДП «УкрНДНЦ», 2016. – 31 с.

8. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – [Чинний від 2007-01-01]. – К. : Держспоживстандарт України, 2007. – 57 с.

9. Programming Guideline [Електронний ресурс] /– Режим доступу: www / URL: https://cache.industry.siemens.com/dl/files/040/90885040/att_970576/v1/81318674_Programming_guideline_DOC_v16_en.pdf.

10. Програмування промислових контролерів Siemens : навч. посіб. / О.В. Зубков. – Харків : ХНУРЕ, 2011. – 122 с.

11. Siemens HMI Design Workbool[Електронний ресурс] /– Режим доступу: www / URL: https://www.awc-inc.com/.

12. Стандарт вищої освіти за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології» галузі знань 15 «Автоматизація та приладобудування» для першого (бакалаврського) рівня вищої освіти [Електронний ресурс] : наказ Міністерства освіти і науки України від 04.10.2018 р. № 1071 // Міністерство освіти і науки України : офіційний вебпортал. – Станом на 18.09.2021. – Режим доступу: https://mon.gov.ua. 41

13. Дипломне проектування для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»: навч. посібник / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова. – Київ-58, пр. Космонавта Комарова, 1, 2016. – 245 с.

14. Методичні вказівки до виконання дипломних проектів (робіт) для студентів напряму підготовки 6.051003 «Приладобудування», професійного спрямування «Прилади і системи орієнтації та навігації»; 151 – Автоматизація та комп'ютерно-інтегровані технології, спеціалізації «Комп'ютерноінтегровані технології і системи навігації та керування»; освітньокваліфікаційного рівня

бакалавр, денної форми навчання, електронне видання / Укл.: П. М. Бондар, В. В. Мелешко. – К.: НТУУ «КПІ», 2016. – 36 с.

15. Технології інформаційно-пошукових систем : навч. посіб. / І. Ш. Невлюдов, А. А. Андрусевич, С. В. Сотник, А. В. Фролов. – Київ : НАУ, 2015. – 336 с. : іл. – ISBN 979-968-577-250-5.

16. Виробничі процеси та обладнання об'єктів автоматизації. Збірник задач : навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, Г. В. Пономарьова, А. О. Функендорф ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Кривий Ріг : КК НАУ, 2018. – 332 с.

17. Невлюдов І. Ш. Технологія програмування промислових контролерів в інтегрованому середовищі CODESYS : навч. посіб. / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2019. – 264 с. : іл. – ISBN 978-966-659- 265-4.

18. Надійність та діагностика приладів та систем : навч. посіб. / В. В. Аврутов, Н. І. Бурау. – Київ : НТУУ «КПІ», 2014. – 156 с.

19. Методичні вказівки з проектної оцінки надійності : навч. посіб. / Ф.М. Андрієв, А.Г. Бердников. – Харків : ХНУ імені В.Н. Каразіна, 2016. – 24 с.

20. Филипенко О. І. Автоматизоване управління технологічним процесом витягування мікроструктурованих оптичних волокон : моногр. / О. І. Филипенко, І. Ш. Невлюдов, Г. В. Пономарьова ; М-во освіти і науки України. – Харків : БУРУН і К, 2015. – 132 с. : іл. – ISBN 978-966-8391-50-7.

21. Автоматизація виробничих процесів : підруч. / І. В. Ельперін, О. М. Пупена, В. М. Сідлецький, С. М. Швед ; М-во освіти і науки України, Нац. ун-т харчових технологій. – 2-е вид., випр. – Київ : Ліра-К, 2016. – 378 с. : іл. – ISBN 978-966-2609-81-3.

22. Кафедра комп’ютерно-інтегрованих технологій, автоматизації та мехатроніки (КІТАМ) [Електронний ресурс] /– Режим доступу: www / URL: https://tapr.nure.ua/.