Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет прикладних інформаційних технологій та електроінженерії
(назва факультету)
Кафедра автоматизації технологічних процесів і виробництв
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: «Розробка навчального комплексу для проектування та дослідження роботи заводського обладнання в програмному пакеті «FACTORY I/O»

Виконав: студент (ка) 6 курсу, групи КАм-61
спеціальності (напряму підготовки) 151
Автоматизація та комп'ютерно інтегровані технології
(шифр і назва спеціальності (напряму підготовки))

(підпис) Сікора Д.А. (прізвище та ініціали)

Керівник (підпис) Шкодзінський О.К. (прізвище та ініціали)

Нормоконтроль (підпис) Козбур І.Р. (прізвище та ініціали)

Рецензент (підпис) Курко А.М. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Робота відноситься до галузі інформаційних технологій, а саме до автоматизації виробничих процесів на основі використання програмованих логічних контролерів для потреб управління технологічним та допоміжним обладнанням.

Метою дипломної роботи є створення навчального комплексу на базі програми імітаційного моделювання для проектування, відлагодження та симуляції роботи заводського обладнання для потреб вищого технічного закладу освіти.

Сфера автоматизації технологічних процесів дуже стрімко розвивається і сприяє мінімізації трудовитрат, підвищенню продуктивності і поліпшенню якості продукції. Сьогодні даному процесу приділяється особлива увага, його застосування дозволяє покласти функції контролю і управління за виробничим процесом на прилади і автоматичні пристрої. Для впровадження засобів автоматизації виробнича галузь потребує кваліфікованих фахівців такого профілю. У зв'язку з цим, тема даної роботи є актуальною.

Розроблений лабораторний комплекс дозволяє створювати імітаційні моделі різноманітних виробничих систем та симулювати роботу заводського обладнання за допомогою програми для тривимірного моделювання. Це дозволяє забезпечити економію витрат університету на забезпечення студентів різноманітним навчальним обладнанням, наочно показати основні принципи роботи фабричного обладнання та особливості їхнього налаштування.

Ключові слова: FACTORY I/O, CONTROL I/O, Codesys, FlexSim, RoboDK, TaraVRbuilder, Simcad, CNC Simulator, PLC, імітаційне моделювання, автоматизація виробництва, симуляція роботи, автоматне програмування.

Магістерська робота містить 138 аркушів пояснювальної записки.

ANNOTATION

The work relates to the field of information technology, namely, the automation of production processes based on the use of programmable logic controllers for the needs of management of technological and auxiliary equipment.

The aim of the thesis is to create a training complex based on a simulation program for designing, debugging and simulating the work of factory equipment for the needs of a higher technical educational establishment.

The field of automation of technological processes is developing very rapidly and helps to minimize labor costs, increase productivity and improve product quality. Today this process is given special attention, its application allows to put the functions of control and management of the production process on devices and automatic devices. In order to implement automation tools, the manufacturing industry requires qualified specialists of such profile. In this regard, the topic of this work is relevant.

The developed laboratory complex allows to create imitation models of various production systems and to simulate the work of factory equipment with the help of the program for its three-dimensional modeling. It allows to save cost of university for providing of students with various educational equipment, to show clearly the basic principles of work of factory equipment and features of their adjustment.

Keyword list: FACTORY I/O, CONTROL I/O, Codesys, FlexSim, RoboDK, TaraVRbuilder, Simcad, CNC Simulator, PLC, simulation modeling, production automation, simulation work, automatic programming.

The master's thesis contains 138 sheets of explanatory note.

ЗМІСТ

ВСТУП

Автоматизація виробництва — вищий рівень розвитку машинної техніки, коли регулювання й керування виробничими процесами здійснюються без участі людини, а лише під її контролем. Сучасний стан розвитку автоматизації виробництва привів до появи якісно нової системи технологічних машин з керуючими засобами, що ґрунтуються на застосуванні електронних обчислювальних машин, програмованих логічних контролерів, інтелектуальних засобів вимірювання і контролю, інформаційно об'єднаних промисловими мережами. Тому з’явилась велика кількість середовищ (наприклад CODESYS, SMLogix від Segnetics, Simatic Step 7 від Simens) для налагодження всіх цих засобів.

На сьогоднішній день, студенти зазнають труднощів в опануванні професії в сфері автоматизації технологічних процесів, без практичної підготовки. На даний момент у сфері автоматизації є великий асортимент обладнання, який просто фізично неможливо надати студентам у повному обсязі для навчання. Розробка лабораторного комплексу в сфері автоматизації, дозволить студентам наочно побачити основні принципи роботи заводського обладнання та їхні тонкощі налаштування. Студенти зможуть працювати з імітаційними моделями різного обладнання, вносити зміни в режимі реального часу та відслідковувати їх в роботі різноманітних заводських систем.

Предметною областю магістерської роботи є імітаційне моделювання. Імітаційне моделювання — це метод дослідження, який ґрунтується на тому, що система, яка вивчається, замінюється імітатором і з ним проводяться експерименти з метою отримання інформації про поведінку цієї системи. Створюється логіко-математичний опис об'єкта, який може бути використаний для експериментування на комп'ютері в цілях проектування, аналізу і оцінки функціонування об'єкта.

Однією із задач дипломної роботи є вибір оптимального програмного середовища та створення лабораторного комплексу для дисципліни «Проектування систем автоматизації». Вибір програми імітаційного моделювання відіграє важливу роль, оскільки напряму впливає на:

- Вибір контролерів, з яким буде функціонувати ПЗ;
- Потужність ПК, на якому буде встановлено ПЗ;
- Вартість ПЗ;
- Функціональні властивості ПЗ.

Наукова та практична значущість розв’язання проблем, притаманних даній спеціальності, яку ставить перед собою дана магістерська робота, полягає у забезпеченні студентам хорошої практичної бази для досягнення високих кількісних і якісних показників при автоматизації технологічних процесів по закінченню університету. Що в свою чергу підвищує продуктивність праці, ритмічність та конкурентоспроможність майбутніх випускників.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Класифікація PLC

1.1.1 Загальні поняття та визначення PLC

Програмований логічний контролер (PLC) – важливий елемент системи автоматизації на промисловому підприємстві. ПЛК необхідні для автоматичного управління об'єктом в умовах реального часу. PLC входить в клас промислових контролерів. У цю групу включені всі технічні засоби, призначені для автоматизації технологічних процесів на виробництві. Основними завданням логічного контролера є збір даних, їх обробка і перетворення, збереження в пам'яті необхідної інформації, створення команд управління, які надходять за допомогою входів і передаються за допомогою виходів. Промислові контролери також використовують для автоматизації будівель, контролю за роботою інженерних мереж і ін. Входи і виходи підключаються до датчиків, механізмів та пристроїв управління. До деяких програмованих логічних контролерів також можна підключити дисплей, миша і клавіатуру.

Логічні контролери здійснюють свою роботу практично без участі оператора, що дозволяє працювати в режимі реального часу в жорстких умовах експлуатації, навіть при наявності несприятливих умов навколишнього середовища.

На зорі розвитку промислової автоматики логічні контролери були створені за типом релейних схем з фіксованою логікою роботи. При порушенні алгоритму доводилося грунтовно змінювати діючу схему. Потім у зв’язку з активним розповсюдженням мікропроцесорної техніки, автоматика виробничого процесу стала будуватися на основі мікропроцесорів.

Сьогодні релейні схеми оснащуються програмним забезпеченням, що перетворює програмовані логічні контролери в мікропроцесорні пристрої, що

забезпечують збір інформації, її переробку, збереження і передачу команд до вузлів виконавчого пристрою.

При цьому програмовані логічні контролери за принципом своєї роботи суттєво відрізняється від мікропроцесорних пристроїв, оскільки програмне забезпечення ПЛК функціонує за аналогією з комп'ютерною операційною системою і забезпечує:

- управління внутрішніми вузлами контролера;
- взаємодія складових компонентів;
- здійснення внутрішньої діагностики.

Системне забезпечення знаходиться в постійній пам'яті процесора і вступає в роботу через кілька мілісекунд після підключення ПЛК до мережі. Програмований логічний контролер працює циклічно, при цьому кожен цикл супроводжується читанням даних і має 4 фази:

- перша являє собою опитування входів;
- на другій фазі здійснюється виконання дій, встановлених користувачем;
- третя фаза встановлює значення входів;
- на четвертій фазі виробляються додаткові операції, наприклад, проводиться діагностика, готуються дані для відладчика, візуалізація.

1.1.2 Класифікація PLC за кількістю точок вводу / виводу [1]

Для класифікації величезної кількості існуючих в даний час контролерів розглянемо їх істотні відмінності. Основним показником ПЛК є кількість каналів вводу-виводу. За цією ознакою ПЛК діляться на наступні групи:

1) Компактні ПЛК – до 320 точок вводу / виводу. Це бюджетні контролери, які є початковими в класі мікрокотролерів. Вони призначені для економічних рішень пов’язаних з автоматизацією виробництва, можуть застосовуватись для управління компактними модульними установками або пристроями, а також як модулі у складі великих складних систем. До відмінних рис даної групи ПЛК можна віднести: простоту, компактність та економічність. Прикладом компактних ПЛК може стати OMRON CP1H зображений на рисунку 1.1.

Рисунок 1.1 PLC OMRON CP1H [1]

2) Модульні ПЛК – до 2560 точок вводу / виводу. Контролери даної групи можна розширювати та встановлювати на DIN-рейку. Такі ПЛК призначені для високошвидкісних завдань, що вимагають високої точності, надійності та багатофункціональності. Широкий набір стандартних модулів введення / виведення (8, 16, 32, 64 точки) і неабиякий набір спеціальних модулів (аналогові, температурні, мережеві, модулі позиціонування) дозволяють найбільш оптимально вирішувати завдання автоматизації, як локальних об'єктів, так і розподілених систем. Використання протоколу MACRO дозволить забезпечити зв'язок з 32 пристроями на кожен порт. Прикладом даної групи ПЛК може стати OMRON CJ2M, який можна побачити на рисунку 1.2.

Рисунок 1.2 PLC OMRON CJ2M [2]

3) ПЛК для стійкового монтажу – до 5120 точок вводу / виводу. Поряд з дуже потужним і зручним програмуванням, контролери даної групи можуть брати на себе додаткові, невластиві контролерам з інших груп, функції, виконувати розширену обробку даних і архівування. Нові інструкції дозволяють обробляти файлову пам’ять, текстові рядки, індексні регістри та багато інших. Дану групу ПЛК можна розширювати до 50 метрів та робити обробку понад 5000 точок вводу / виводу. Широкий набір мережевих модулів і наднизький час виконання базових інструкцій (0.04 мкс) робить дану групу ПЛК найпотужнішими в сегменті. Прикладом може стати OMRON CS1D зображений на рисунку 1.3.

Рисунок 1.3 PLC OMRON CS1D [3]

1.1.3 Класифікація PLC за структурою [4]

За конструктивним виконанням ПЛК поділяють на два типи: моноблочні (рис. 1.4) і модульні. У корпусі моноблочного ПЛК поряд з ЦП, пам'яттю і блоком живлення розміщується фіксований набір входів / виходів.

Рисунок 1.4 Моноблочні програмовані логічні контролери [4]

У модульних ПЛК використовують окремо встановлені модулі входів / виходів. Згідно з вимогами IEC 61131, їх тип і кількість можуть змінюватися в залежності від поставленого завдання і оновлюватися з часом. ПЛК подібної концепції представлені на рисунку 1.5. Подібні ПЛК можна розширювати через інтерфейс Ethernet.

Рисунок 1.5 Модульні програмовані логічні контролери [4]

Моноблочні функціонально завершені ПЛК можуть включати в себе невеликий дисплей і кнопки управління. Дисплей призначений для відображення поточних робочих параметрів і введених за допомогою кнопок команд робочих програм і технологічних установок. Більш складні ПЛК комбінуються з окремих функціональних модулів та спільно закріплюються на стандартній монтажній рейці. Залежно від кількості обслуговуваних входів і виходів, встановлюється необхідна кількість модулів введення і виведення.

Джерело живлення може бути вбудованим в основний блок ПЛК, але зазвичай його виконують у вигляді окремого блоку живлення (БП), що закріплюється поруч на стандартній рейці.

Первинним джерелом для БП найчастіше служить промислова мережа 24/48/110/220/400 В, 50 Гц. Інші моделі БП можуть використовувати в якості первинного джерело постійної напруги 24/48/125 В. Стандартними для промислового обладнання та ПЛК є вихідні напруги БП: 12, 24 і 48 В. У системах підвищеної надійності можлива установка двох спеціальних резервних БП для дублювання електроживлення .

1.1.4 Класифікація PLC за виконуваними функціями [4]

Відповідно до різних функцій ПЛК можна розділити на три групи: низького, середнього, високого класу.

1) ПЛК низького класу мають такі базові функції, як: логічні операції, підрахунок, переміщення, самодіагностика та моніторинг. Також вони можуть мати арифметичної операції, комунікаційні функції, передачу даних та порівняння. В основному вони використовується для логічного управління та підтримання певної послідовності дій.

2) ПЛК середнього класу. На додаток до функцій низькорівневих програмованих логічних контролерів, ПЛК середнього рівня мають сильний аналоговий вхід / вихід та можуть здійснювати операції перетворення сигналу (цифровий в аналоговий, аналоговий в цифровий). Даний клас ПЛК також

підтримує віддалене керування, управління перериваннями, ПІД-регулювання та інші функції, які застосовуються до складних систем управління.

3) ПЛК високого класу. На додаток до функцій ПЛК середнього рівня, PLC високого класу підтримують матричні операції, операції бітової логіки, квадратні кореневі операції, функції табуляції та функції передачі в таблиці. Також, даний клас ПЛК має більш міцну функцію комунікації, яка може бути використана для великомасштабного управління процесом та являє собою систему розподіленої мережі, що забезпечує автоматизацію заводу.

1.2 Огляд програм імітаційного моделювання роботи фабричного обладнання

Одним із методів вирішення проблем, які виникають при створенні та розробці автоматизованих систем є імітаційне моделювання. Імітаційне моделювання – це метод дослідження, при якому досліджувана система замінюється моделлю, що з достатньою точністю описує реальну систему, з нею проводяться експерименти з метою одержання інформації про цю систему. На даний момент існує безліч програм за допомогою яких можна моделювати, кастомізувати та відтворювати у часі різноманітні технологічні процеси за допомогою 3D анімації.

Для створення лабораторного комплексу потрібно обрати програму імітаційного моделювання фабричного обладнання, яка буде працювати з навчальними контролерами. Даний тип програм призначений для симуляції трьохвимірного моделювання різноманітних виробничих процесів. На даний час такі програми стрімко набувають популярність. Це зв’язано з тим що теперішні технології дають змогу досить реалістично відображати різноманітні процеси без використання надпотужностей ЕОМ.

Найпопулярнішими програмами імітаційного моделювання на сьогоднішній день є:

- FACTORY I/O;
- FlexSim;
- RoboDK;
- TaraVRbuilder;
- Simcad;
- CNC Simulator.

Усі вищезгадані програмні засоби будуть розглянуті у даній дипломній роботі, розібрані та порівняні.

1.2.1 Factory I/O [5]

FACTORY I/O (рис. 1.6) – це програма для трьохвимірного фабричного моделювання та вивчення технологій автоматизації. Вона створена таким чином, щоб бути простою у використанні, що дозволяє швидко побудувати віртуальну фабрику за допомогою загальних промислових частин. FACTORY I/O також включає в себе безліч сцен, починаючи від початкових до складніших рівнів.

Найчастіше FACTORY I/O використовують, як навчальну платформу для PLC, оскільки PLC є найбільш поширеними контролерами у промислових програмах. Однак, її також можна використовувати та підключати до мікроконтролерів або будь-яких інших пристроїв через інтерфейсні плати, наприклад Advantech USB 4750/4704. Це плати DAQ (системи збору даних), що працюють через USB та які можна використовувати на настільних, портативних та планшетних ПК до тих пір, поки USB порт є доступним. Зовнішнє джерело живлення не потрібне. Функціональними особливостями програми FACTORY I/O є те, що вона працює з усіма брендами PLC та найпоширенішими технологіями автоматизації. Програма використовує драйвери для взаємодії з PLC, SoftPLC, симуляторами PLC, Modbus TCP/IP, OPC Client DA/UA та багатьма іншими технологіями.

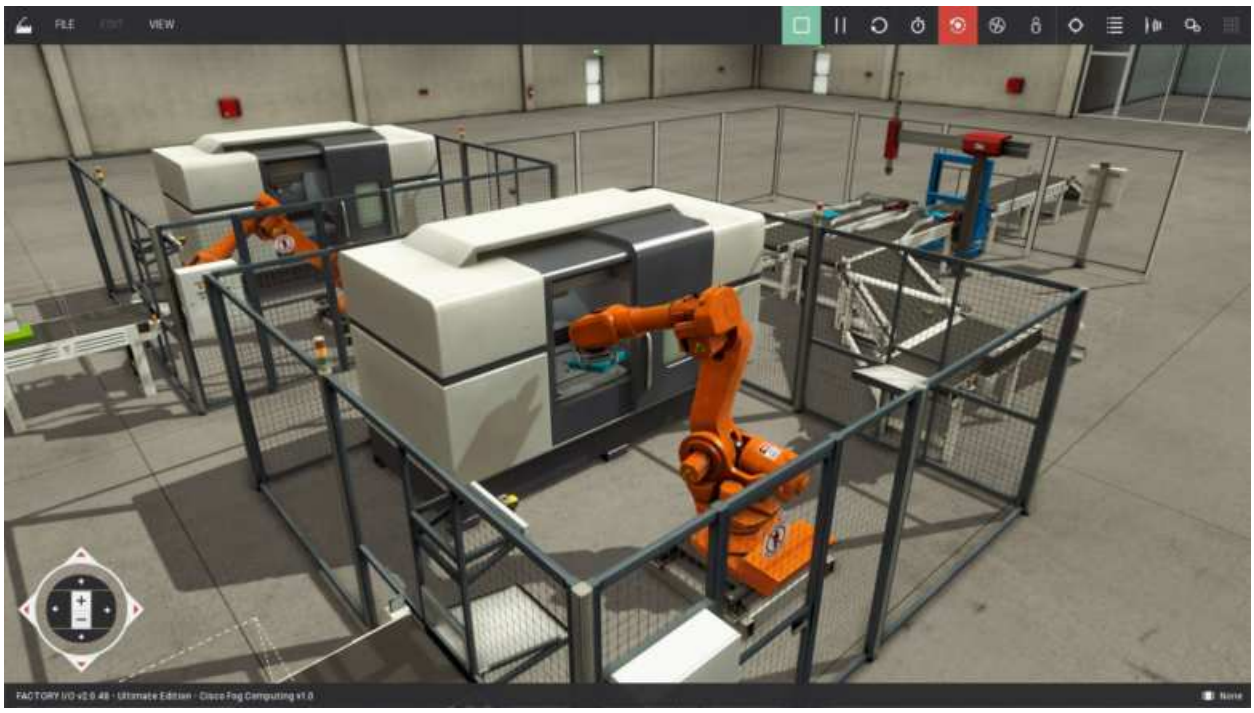


Рисунок 1.6 Вікно програми FACTORY I/O

Кожне видання даної програми містить пакет драйверів для конкретної технології (наприклад, Allen-Bradley Edition, Siemens Edition, Automgen Edition).

Програма надає можливість практикувати реальні контрольні задачі з використанням більш ніж 20 сцен. Також можливим є створення власних сцен з використанням більш ніж 80 промислових частин. Присутня можливість використовувати цифрове значення для запуску або зупинки конвеєра, аналогового значення для зважування предметів або регулювання рівня рідини. Перевагами програми FACTORY I/O є:

- Легка інтеграція з існуючим навчальним обладнанням
- Взаємодія**.** Можливість інтегрування CODESYS разом з FACTORY I/O через OPC Data Access.
- Відкритість системи. FACTORY SDK (Software Development Kit) надає інструменти, щоб розробники могли створювати власні програми, що мають доступ до симуляційних точок вводу-виводу.

Системні вимоги для FACTORY I/O для ПК зображено у таблиці 1.1.

Таблиця 1.1 – Системні вимоги FACTORY I/O

Операційна система	Windows Vista або вище
Процесор	Intel Core 2 Duo на 2 ГГц або AMD Athlon 64 x2 2 ГГц або вище.
Пам'ять	1 Гб
Простір на жорсткому диску	600 Мб
Відеокарта	NVIDIA з 2007 р. (GeForce 8 Series), картки AMD з 2007 р. (Radeon 2xxx Series), карти Intel з 2008 р. (GMA 4500); підтримка шейдерної версії 2.0 або вище.
Звук	DirectX сумісна звукова карта
DirectX	9.0c
Інше	.NET Framework 4.5

1.2.2 FlexSim [6]

FlexSim (рис. 1.7) - це програмне забезпечення для 3D-моделювання, яке імітує, прогнозує та візуалізує системи у виробництві, обробці матеріалів, охороні здоров'я, складанні, видобутку, логістиці тощо. FlexSim допомагає оптимізувати поточні та заплановані процеси, визначати та зменшувати витрати, зменшувати витрати та збільшувати доходи. Програма дає можливість імітувати не тільки поведінку системи з середини, але і динамічно змінювати зовнішній вигляд, що дозволяє відслідковувати весь процес та спостерігати за тим що відбувається.

За допомогою ІМ (Імітаційного моделювання) в системі FlexSim можна визначити пропускні спроможності підприємств, збалансованість виробничих ліній, виявити вузькі місця виробничих процесів, перевірити нові методи планування випуску продукції, оптимізувати виробничі процеси, обґрунтувати капіталовкладення.

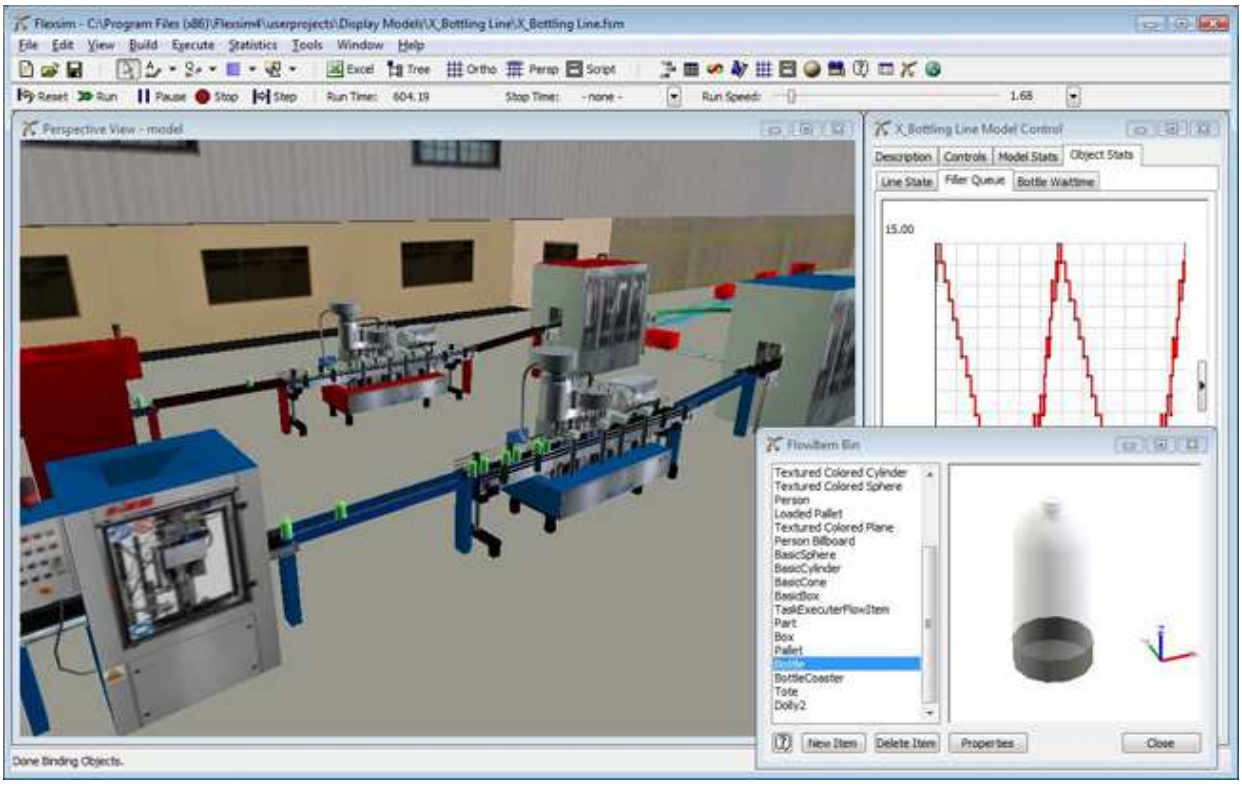


Рисунок 1.7 Вікно програми FlexSim

Кожна модель в системі Flexsim може бути представлена у тривимірній віртуальній реальності. Система Flexsim надає можливості для створення моделей і підмоделей безпосередньо на мові C ++, на якій в системі проводиться перехід від структурної до імітаційної моделі. В системі FlexSim є потужна система 3D-графіки, що дозволяє створювати діаграми і графіки для динамічного відображення стану об'єкта моделювання. Перевагами програми FlexSim є:

- Реалізм. Програмне забезпечення для моделювання FlexSim використовує OpenGL, таку ж графічну бібліотеку, яка використовується в сучасних 3D-іграх.
- Статистика. Потужна 3D-графіка FlexSim дозволяє в моделях графіків і діаграм динамічно відображати вихідну статистику.
- Взаємодія. Можливість імпорту креслень з різноманітних 2D та 3D CAD програм, щоб використовувати їх як поверхневі плани або макети.

Системні вимоги до програми FlexSim для ПК зображено у таблиці 1.2.

Таблиця 1.2 – Системні вимоги FlexSim

ЦП	Будь-який сучасний процесор Intel або AMD
ОЗП	4 Гб оперативної пам'яті
Графіка	Будь-який графічний процесор, який підтримує OpenGL 3.1 або новішої версії: • Nvidia GeForce 300 Series або вище • AMD Radeon R600 (HD 2xxx, HD 3xxx) • Intel HD 2000 або вище
ОС	Будь-яка підтримувана операційна система Microsoft Windows. На даний момент це - Windows 8, 8.1 і 10.
Архітектура	32 або 64 біт
Додаткове програмне забезпечення	.NET Framework (обов'язково) Visual C ++ (необов'язково)

1.2.3 RoboDK [7]

RoboDK (рис. 1.8) – це сучасна програма для моделювання та автономного програмування промислових роботів. Програмування в автономному режимі означає, що програми для роботів можуть бути створені, імітовані та реалізовані в режимі офлайн. Програмне забезпечення може використовуватися для багатьох виробничих проектів: фрезерування, зварювання, збирання, розміщення, пакування, маркування, палетування, фарбування, калібрування та багатьох інших. Програма дозволяє програмувати роботів, як на мові Python так і на C#.

Функціональними особливостями програми RoboDK є те, що вона забезпечує зручний графічний інтерфейс користувача, який є досить простим, що дозволяє легко моделювати та програмувати промислові роботи без глибоких знань у програмуванні. RoboDK має бібліотеку, яка складається з понад 200 роботів від 15 різних виробників, включаючи ABB, Fanuc, Kuka та Motoman.

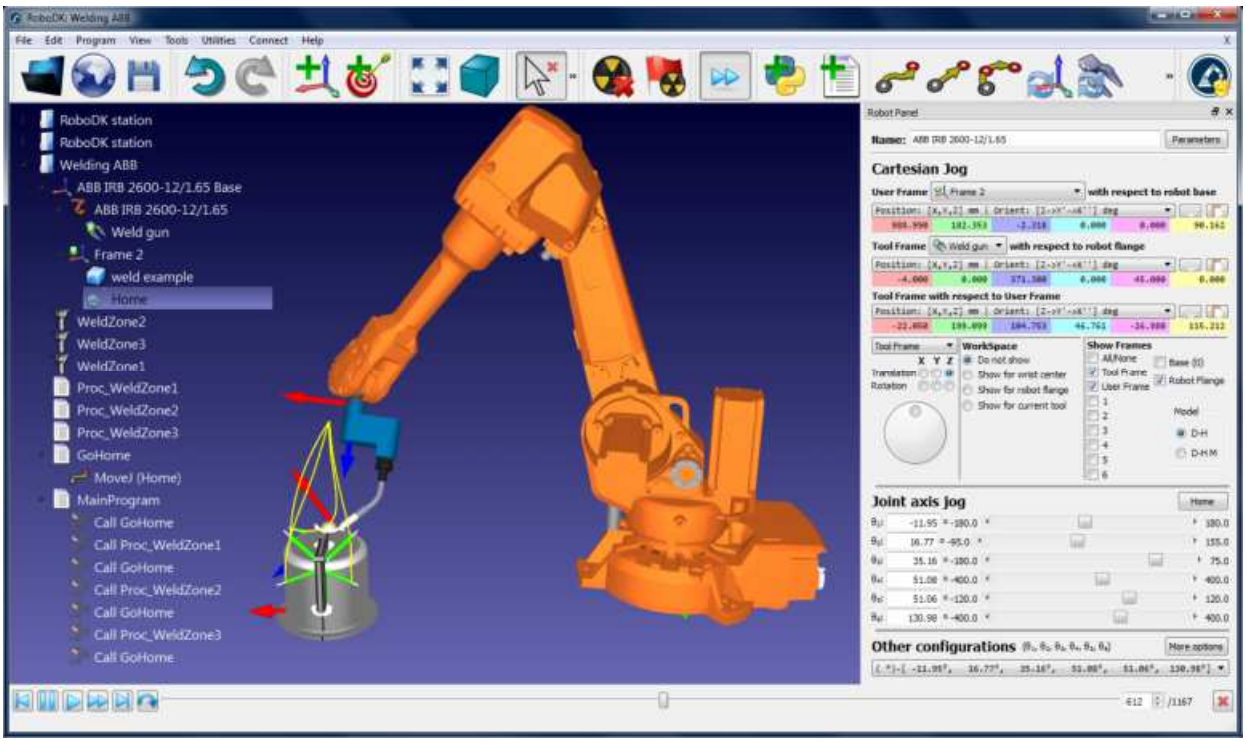


Рисунок 1.8 Вікно програми RoboDK

Додатки RoboDK дозволяють експортувати програми до справжнього робота, включаючи ABB Rapid (mod/prg), Fanuc LS (LS/TP), Kuka KRC/IIWA (SRC/java), Motoman Inform (JBI), Universal Robots (urscript). Також дана програма дозволяє калібрувати справжні роботи. За допомогою API RoboDK можна програмувати та імітувати роботу роботів використовуючи мову програмування Python. Вона дозволяє працювати швидше та інтегрувати системи більш ефективно. Python дозволяє виражати концепції меншою кількістю рядків коду в порівнянні з іншими мовами. Перевагами RoboDK є:

- Багатоплатформенність. Можливість працювати на різноманітних операційних системах (Windows, MacOS, Linux, Android).
- Взаємодія. Можна імпортувати різні типи файлів, включаючи файли step та iges.
- Безкоштовність. Програма є повністю безкоштовною та не потребує підключення до інтернету, що дає повну автономність.

Системні вимоги до програми RoboDK для ПК зображено у таблиці 1.3.

Таблиця 1.3 – Системні вимоги RoboDK

Операційна система	Windows, MacOS, Linux, Android
Процесор	Будь-який сучасний процесор Intel або AMD
Пам'ять	1 Гб
Простір на жорсткому диску	100 Мб
Відеокарта	Будь-яка сучасна відеокарта
Звук	DirectX сумісна звукова карта
DirectX	9.0c
Інше	.NET Framework 4.5

1.2.4 TaraVRbuilder [8]

TaraVRbuilder (Рис. 1.9) - це програмний засіб для легкого та швидкого створення різноманітних динамічних систем та відображення їх в логістичних та заводських процесах.

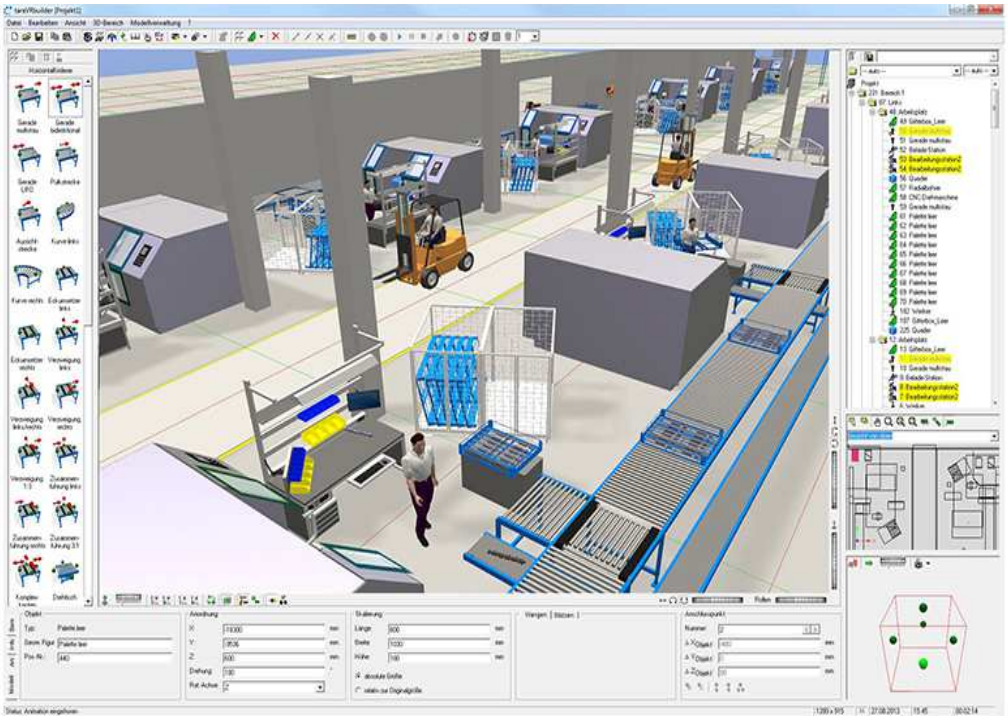


Рисунок 1.9 Вікно програми TaraVRbuilder

Окрім різноманітних розгалужених технологій транспортування та виробництва, також доступно багато виробничих товарів та елементів, які підлягають взаємодії та транспортуванню на конвеєрах. TaraVRbuilder дозволяє створювати симуляцію та анімацію різноманітних 3D об’єктів та комбінувати їх з різними змінними процесами.

Користувачі без досвіду роботи у CAD системах та програмування можуть швидко та легко створювати 3D сцени. У програмі присутня просто величезна бібліотека фабричних та логістичних елементів, яка складаються з понад 500 об'єктів і дозволяє швидкого та легко налаштовувати систему для власних потреб. В бібліотеках TaraVRbuilder можна зустріти такі елементи:

- Конвеєрні системи, портали та роботи;
- Транспортні засоби, навантажувачі;
- Стелажі та пристрої для зберігання;
- Штабелеві крани, пересувні стелажі;
- Стовпи, стіни, майданчики, доріжки та сходи;
- Огорожі, будівлі, пандуси та ворота;
- Шафи, столи, комп’ютери тощо;
- Товари для транспортування, деталі машин;

Також в програмі є потужна система, що дозволяє створювати діаграми і графіки для динамічного відображення стану об'єкта моделювання та виводити їх у форматі Excel. Перевагами TaraVRbuilder є:

- Можливість імпортування даних 2D або 3D (наприклад, макет, план поверху, будівництво)
- Простота. Низькі інвестиційні та кваліфікаційні пороги. Непотрібно знати мови програмування. Вибір та позиціонування за допомогою «перетягування».
- Відео. Просте отримання відеороликів. Екскурсії та огляди через віртуальну систему.

Системні вимоги до програми TaraVRbuilder для ПК зображено у таблиці 1.4.

Таблиця 1.4 – Системні вимоги TaraVRbuilder

Операційна система	Windows 7 або вище
Процесор	Будь-який сучасний процесор Intel або AMD
Пам'ять	1 Гб
Простір на жорсткому диску	500 Мб
Відеокарта	Будь-яка сучасна відеокарта
Звук	DirectX сумісна звукова карта
DirectX	9.0c
Інше	.NET Framework 4.5

1.2.5 Simcad [9]

Simcad (рис. 1.10) є продуктом компанії CreateASoft, який використовується для моделювання різноманітних середовищ, таких як виробництво, постачання, логістика, дистрибуція, охорона здоров'я та багато інших. Програмне забезпечення являє собою інструмент для планування, автоматизації, організації, оптимізації та інжинірингу реальних технологій та процедур. Simcad дозволяє створити комп'ютерну модель у реальному середовищі, якою може керувати користувач та вносити зміни, коли модель працює, для цілком реалістичного моделювання. Використовуючи модель, можна перевірити ефективність будь-якого можливого сценарію, а також знайти точки поліпшення та оптимізації виробничого процесу.

Simcad має зручний графічний інтерфейс для побудови імітаційної моделі. Вміння програмувати для того щоб побудувати модель зовсім непотрібні. Вся модель може бути побудована і відредагована з допомогою графічно-призначеного інтерфейсу, в тому числі для будь-яких передових характеристик імітаційної моделі.

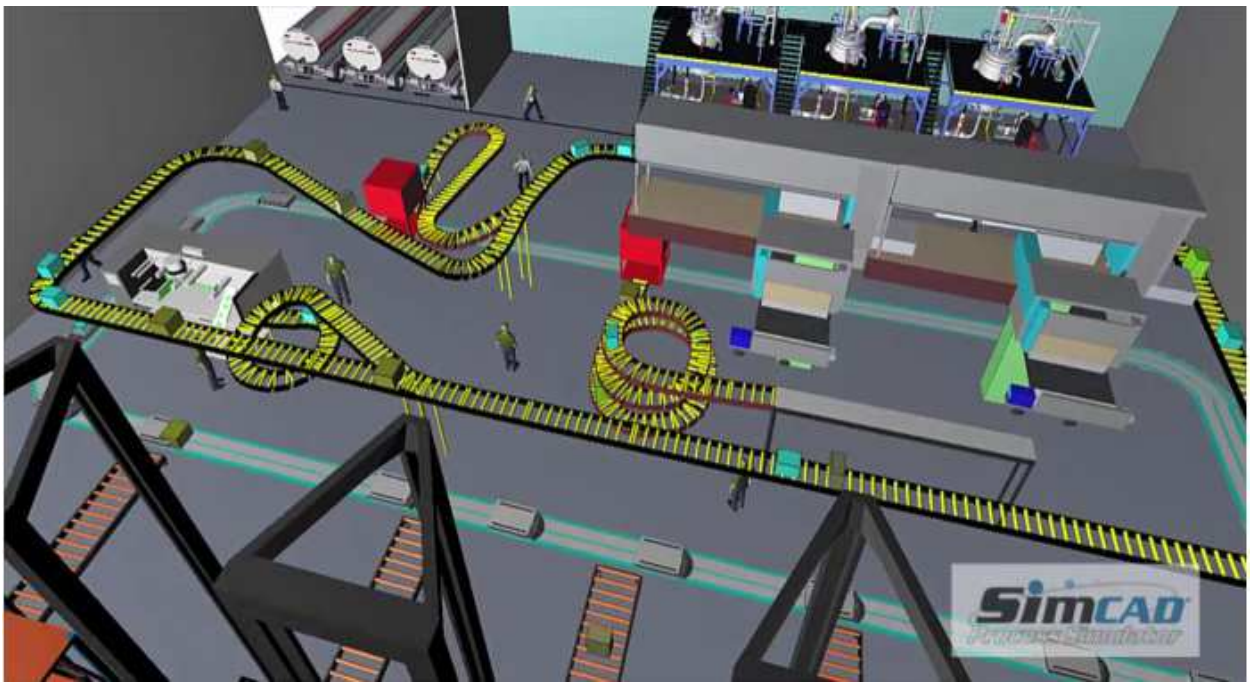


Рисунок 1.10 Вікно програми SimCad

Крім того, Simcad дозволяє імпортувати дані майже з будь-якого джерела даних, включаючи бази даних Microsoft Access , Excel , Visio та SQL Server для спрощення створення моделі. Також Simcad підтримує відстеження в реальному часі, використовуючи пристрої, включаючи RFID та сканер штрих-кодів . Симуляцію виробничих процесів можна переглядати як у 2D, так і в 3D-візуалізаціях. Перевагами програмного забезпечення Simcad є:

- Динамічність. Можливість вносити зміни до моделі під час виконання симуляції.
- Прогнозування. Відслідковування та передбачення можливих подій, впливатимуть на операції: зміни в попиті, перебої в постачанні, зміни в товарній комбінації тощо.
- Звітність. Отримання інтегрованих звітів по створених сценаріях. Можливість проаналізувати відмінність та ефективність ручної праці від автоматизації

Системні вимоги до програми Simcad для ПК зображено у таблиці 1.5.

Таблиця 1.5 – Системні вимоги Simcad

Операційна система	Windows 7 SP1 або вище
Процесор	Будь-який сучасний процесор Intel або AMD
Пам'ять	4 Гб
Простір на жорсткому диску	10 Гб
Відеокарта	Будь-який сучасна відеокарта
Звук	DirectX сумісна звукова карта
DirectX	9.0c
Інше	.NET Framework 4.5

1.2.6 CNC Simulator [10]

CNCSimulator (рис. 1.11) – це симулятор числового програмного керування. Але дана програма являється не тільки симулятором ЧПК, але й містить систему CAD/CAM під назвою SimCam, 3D-модель фрезерного інструменту під назвою 3D Maker, інструмент передач - Gear Maker, і багато інших інструментів і функцій, для програмістів верстатів ЧПК, любителів, студентів і викладачів.

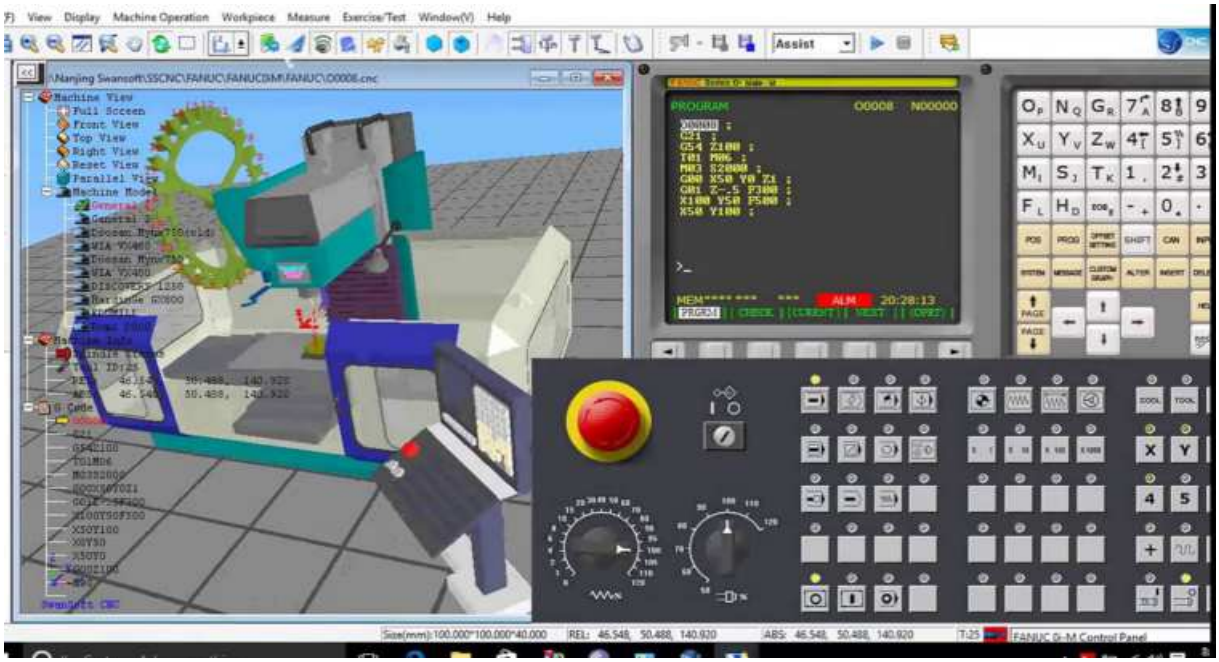


Рисунок 1.11 Вікно програми CNCSimulator

Програма підтримує повну симуляцію та налагодження роботи верстата. Програмування здійснюється за допомогою G-коду. CNCSimulator підтримує токарні і фрезерувальні групи верстатів.

CNCSimulator дозволяє здійснювати симуляцію всіх етапів процесу: вибір заготовки, обнулення заготовки, вибір і вимірювання інструменту, вибір правильного режиму роботи верстата на панелі управління. Симуляція включає в себе реалістичні компоненти, такі як: охолоджуюча рідина, звуки механічної обробки, утворення стружки. Можливість настройки робочих нулів, корекції інструменту і використання різних затискних пристосувань. Первагами програмного комплексу CNCSimulator є:

- Реалістичність. 3D-Моделювання засноване на OpenGL, що дозволяє швидко і точно створювати 3D анімацію.
- Відео. Можливість запису відео роликів і збереження в форматі AVI.
- Бібліотека. Величезна бібліотека для вибору верстатів, метеріалів заготовки, типів інструментів та можливість створювати власні.

Системні вимоги до програмного комплексу CNCSimulator для ПК зображено у таблиці 1.6.

Таблиця 1.6 – Системні вимоги CNCSimulator

Операційна система	Windows 7 або вище
Процесор	Intel Pentium і вище
Пам'ять	1 Гб
Простір на жорсткому диску	500 Мб
Відеокарта	NVIDIA GeForce, ATI Radeon або аналогічна відеокарта
DirectX	9.0c
Інше	Доступ до мережі інтернет

1.3 Висновок

Проведений огляд наявних програм імітаційного моделювання, дозволяє зробити висновок, що вони відображають ряд характерних тенденцій у розвитку симуляційного програмного забезпечення. Відібрані приклади програм відображають багатогранність даної сфери, та показують що для різних задач можна користуватись характерно різним програмним забезпеченням. Слід зазначити одну важливу особливість програм подібного роду. Незважаючи на досить велику вартість даних програмних засобів та невелике розповсюдження, вони користуються великим попитом у вузькоспеціалізованих сферах.

На основі розглянутих контролерів, потрібно обрати програму, яка буде забезпечувати взаємодію з контролерами або їх симуляторами. Коли стоїть задача для оптимізації виробництва, краще скористатись можливостями програмного засобу FlexSim. Він краще адаптований під такі задачі та має вмонтовані функції, які відсутні в інших програмних продуктах. В програмах FlexSim, TaraVRbuilder, Simcad відсутня можливість для програмування PLC та подальше їх налагодження. Тому для розробки систем автоматизації краще скористатись програмним пакетом FACTORY I/O, в якому можна здійснювати програмування контролерів, як зовнішніми засобами (CODESYS), так і інтегрованими рішеннями (CONTROL I/O). Але в порівнянні з FACTORY I/O, TaraVRbuilder має кращу кастомізацію, через наявність більшої кількості об’єктів у бібліотеці. Тому при потребі створення повноцінних динамічних та логістичних систем краще скористатись нею. Simcad Pro краще покаже себе при аналітиці, відслідковуванні та передбаченні можливих подій, які впливатимуть на перебіг процесу: зміни в попиті, перебої в постачанні, зміни в товарній комбінації тощо. Спеціалізація програмного засобу RoboDK випливає з назви, він найкраще справиться із завданнями пов’язаними з імітацією роботи роботів або їх моделюванням. А коли стоїть задача симуляції роботи верстатів та всіх пов’язаних з ним процесів, тут краще себе проявить CNCSimulator.

Отже, проведений аналіз літературних джерел свідчить про те, що для розробки навчального комплексу найкраще підходить програма FACTORY I/O, на

основі якою будуть розроблятись лабораторні роботи. Вибір програми FACTORY I/O зумовлено тим, що вона в деякій мірі є монополістом у даній ніші програмних комплексів. Серед великої кількості програмних засобів доступних на даний момент, тільки FACTORY I/O дозволяє одночасно і програмувати контролери і здійснювати імітацію їхньої роботи за допомогою 3D-візуалізації. Для повноцінної роботи з FACTORY I/O абсолютно не обов'язково мати справжній контролер. У функціоналі даної програми є симулятори ПЛК, які можна запрограмувати за допомогою середовища CODESYS, яке вивчається студентами на інших дисциплінах кафедри автоматизації технологічних процесів та виробництв. У бібліотеці FACTORY I/O є величезна колекція деталей на основі найпоширенішого промислового обладнання (конвеєри, навантажувальні пристрої, датчики, ліфти). Також, переважаючим критерієм при виборі FACTORY I/O стало те, що студенти зможуть ознайомитись з даним програмним продуктом у домашніх умовах використовуючи безкоштовну 30 денну пробну версію.

2 НАУКОВО–ДОСЛІДНА ЧАСТИНА

2.1 Процедури розробки автоматизованих систем

Найголовніші інженерні задачі, що виникають при розробці СА (систем автоматизації), вирішуються під час створення та проектування схем автоматизації. В автоматизованих системах об’єктами керування є сукупність основного та допоміжного устаткування, а також потоки сировини, енергії та інших матеріалів, що визначають особливості технологічного процесу. Вирішуючи основні задачі керування технологічним процесом, АСКТП можуть впливати на показники діяльності підприємства таким чином: збільшити об’єм виробництва на тих самих виробничих площах і при тих самих потужностях, пом’якшити або повністю усунути аритмічність в роботі обладнання цехів і підприємства, що автоматизується в цілому, підвищити якість продукції, скоротити або ліквідувати брак у виробництві, збільшити завантаження обладнання за потужністю і часом, скоротити втрати матеріальних ресурсів і знизити питомі норми їх витрачення, вивільнити оборотні кошти, ліквідувувати непродуктивні витрати, скоротити чисельність персоналу. [12, с.54]

Можна виділити таку послідовність вирішення основних інженерних завдань, які виникають при розробці схем автоматизації:

2.1.1 Аналіз технологічного процесу

Розроблення ефективної СА можливе тільки на засадах глибоких знань технології виробництва, що автоматизується, його технологічного регламенту, конструктивних особливостей технологічного устаткування та режимів його роботи. Як правило, об’єктом автоматизації є технологічний комплекс (ТК), тому при цьому аналізуються:

а) Тип виробництва, в яке входить ТК: неперервне, дискретне або неперервно-дискретне;

б) Тип структури ТК: паралельна, послідовна та різні модифікації цих структур.

в) Тип апаратів, з яких складається ТК: неперервні, періодичні або напівнеперервні, та які типові технологічні процеси в них відбуваються;

г) Наявність буферних проміжних ємностей на початку і в кінці ТК.

Вихідні дані для проектування одержують на підготовчому етапі (під час обстеження підприємства) і в процесі проектування. Перелік вихідних даних значною мірою залежить від об'єкту автоматизації. [11, с.62]

2.1.2 Визначення інформації, необхідної для оцінки стану об'єкта

Більшість об'єктів, які автоматизуються на сьогоднішній день, є об'єктами з розподіленими параметрами, і тому велика увага приділяється вибору точок для отримання інформації. Під час проектування СА, необхідно оприділитись з величинами, інформація про які буде збиратись та досліджуватись для оцінювання стану об'єкта автоматизації. Детальне вивчення об'єкта автоматизації і необхідні науково-дослідні роботи, пов'язані з пошуком величин та інформації, мають велике значення для подальшої побудови СА, тому що надлишкова інформація суттєво збільшує вартість системи, а недостатня може призвести до помилок у реалізації технологічного процесу.

2.1.3 Аналіз існуючих систем автоматизації

Кінцеве рішення про автоматизацію виробництва приймають після аналізу існуючих вітчизняних та закордонних систем автоматизації. Потрібно зробити детальне обстеження систем та об'єктів що вже є на ринку. Основна мета цього етапу полягає у виявленні головних передбачуваних джерел ефективності створюваної АСКТП.

Новостворений варіант СА повинен враховувати досвід розроблення аналогічних систем і повинен забезпечити досягнення найкращих техніко-економічних показників серед існуючих СА подібних об'єктів. При цьому він

повинен враховувати можливі перспективи модернізації систем автоматизації і необхідності вирішувати більш складні алгоритми управління.

Методично етап зводиться до ретельного вивчення й аналізу діючих систем і об’єкту керування. Виявляються існуючі недоліки, що призводять до зменшення ефективності виробництва, а також визначаються причини цих недоліків. Обстеження проводиться шляхом збору і вивчення відповідних матеріалів, досвіду працюючого персоналу і безпосереднім оглядом. Якщо проект розробляється для новоспоруджуваного підприємства, необхідно обстежувати підприємства-аналоги. [12, с.59]

2.1.4 Вибір каналів регулюючих дій

Ця задача вирішується за допомогою вибору вихідних величин каналів, притримуючись при цьому технологічного регламенту та сформованої мети керування об’єктом. Потім аналізуючи основні характеристики можливих каналів регулювальних дій (статичні та динамічні), можна вибрати ті, які краще підходять для реалізації керування та відповідають технологічному регламенту. Зазвичай перевагу мають канали, які мають невеликі інерційності та запізнення.

Коефіцієнти передачі треба підбирати так, щоб вони дозволяли мінімізувати вплив на відповідну вихідну величину найбільшого із збурень. Такий підхід використовують у разі автоматизації діючих об’єктів. Під час автоматизації нових об’єктів доцільно одночасно створювати технологічний процес та систему управління ним. У такому випадку вони будуть найкраще відповідати вимогам до автоматизованих процесів. Тому на стадії проектування спеціалісти з автоматизації будуть задавати, а спеціалісти-технологи реалізовувати необхідні статичні та динамічні властивості об’єкта. Така послідовність дає змогу отримати необхідну якість процесу керування. Під час вирішення подібних задач, також широко застосовуються системи автоматизації типових технологічних процесів. [11, с.62]

2.1.5 Вибір на технологічній схемі інформаційних точок

На даний момент основна кількість об’єктів, які автоматизують, являються об’єктами з розподіленими параметрами. Від коректного вибору точок для зняття інформації залежить не тільки динамічна та статична точність оцінки параметрів, а й достовірність інформації про досліджуваний об’єкт. Дуже великим фактором є те, що місце розташування точок зняття інформації впливає на характеристики об’єкту (динамічні та статичні) і відповідно на параметри регулювання регулятора, показники функціонування системи та структуру системи.

Зазвичай, технологічна величина про яку збирається інформація, є розподіленою за довжиною агрегату і неперервно змінюється у часі. Тому, основним завданням, яке витікає при постановці задачі, є вибір місця розташування точок та їх кількість. Відстань, яка необхідна між інформаційними точками знаходиться за допомогою алгоритму інтерполяції, задаючи середню квадратичну похибку апроксимації функції розподілу відповідної величини за довжиною агрегату. [11, с.63]

2.1.6 Вибір технічних засобів

Цей пункт є одним з найскладніших та найважливіших на етапі розроблення систем автоматизації. Якщо будувати такі системи використовуючи мікропроцесорну техніку, то можна виділити такі основні під задачі:

- Вибір МПК та їх кількості;
- Розрахунок апаратного складу МПК;
- Вибір засобів для отримання інформації;
- Вибір засобів для подання інформації;
- Вибір засобів для реалізації регулюючих дій.

При використанні дисплея, як засоба для подання інформації, потрібна додаткова підзадача вибору персонального комп’ютера. Вибираючи всі ці засоби необхідно,

щоб вони відповідали стандартам, були серійними та забезпечували відкритість системи. [11, с.63]

2.1.7 Вибір місця розміщення технічних засобів

На основі функціональної схеми та вибраної комплектації засобів управління створюється структурна схема для комплексу технічних засобів. Сучасні автоматизовані системи керування є ієрархічними системами і мають три рівні.

На нижньому рівні знаходяться модулі віддаленого вводу-виводу, первинні перетворювачі, виконавчі механізми з регулюючими органами і прилади розташовані по місцю. На середньому рівні зазвичай знаходяться програмовані логічні контролери. А на верхньому рівні управління розміщуються промислові або персональні комп'ютери та панелі оператора. Для зв'язку верхнього рівня із середнім використовується з'єднання «точка-точка» (PtP) або локальна промислова мережа. З'єднання «точка-точка» застосовуються в простих випадках, наприклад, для з'єднання одного програмованого контролера з комп'ютером. [13, с.13]

2.1.8 Складання алгоритмів схем управління

Невід’мним етапом процесу проектування систем управління є створення та оптимізація алгоритмів управління. Необхідно створити загальний алгоритм роботи, який буде відображати та пояснювати взаємодію всіх рівнів управління враховуючи їх особливості. Також потрібно відобразити взаємодію оперативного персоналу з системою управління. Весь алгоритм роботи складається узагальнено. Деякі режими роботи можна показувати у вигляді конкретних процесів. Паралельними лініями відображати ті процеси, які виконуються одночасно.

Весь алгоритм керування можна доповнювати окремими алгоритмами для окремих завдань. Найважливішими алгоритмами є алгоритми управління

об’єктами на основі контролерів. Такі алгоритми можна розбивати на окремі завдання та робити їх подальшу деталізацію. [13, с.15]

2.2 Висновок

Структурно-логічну схему процедури розробки автоматизованих систем можна побачити на рисунку 2.1.

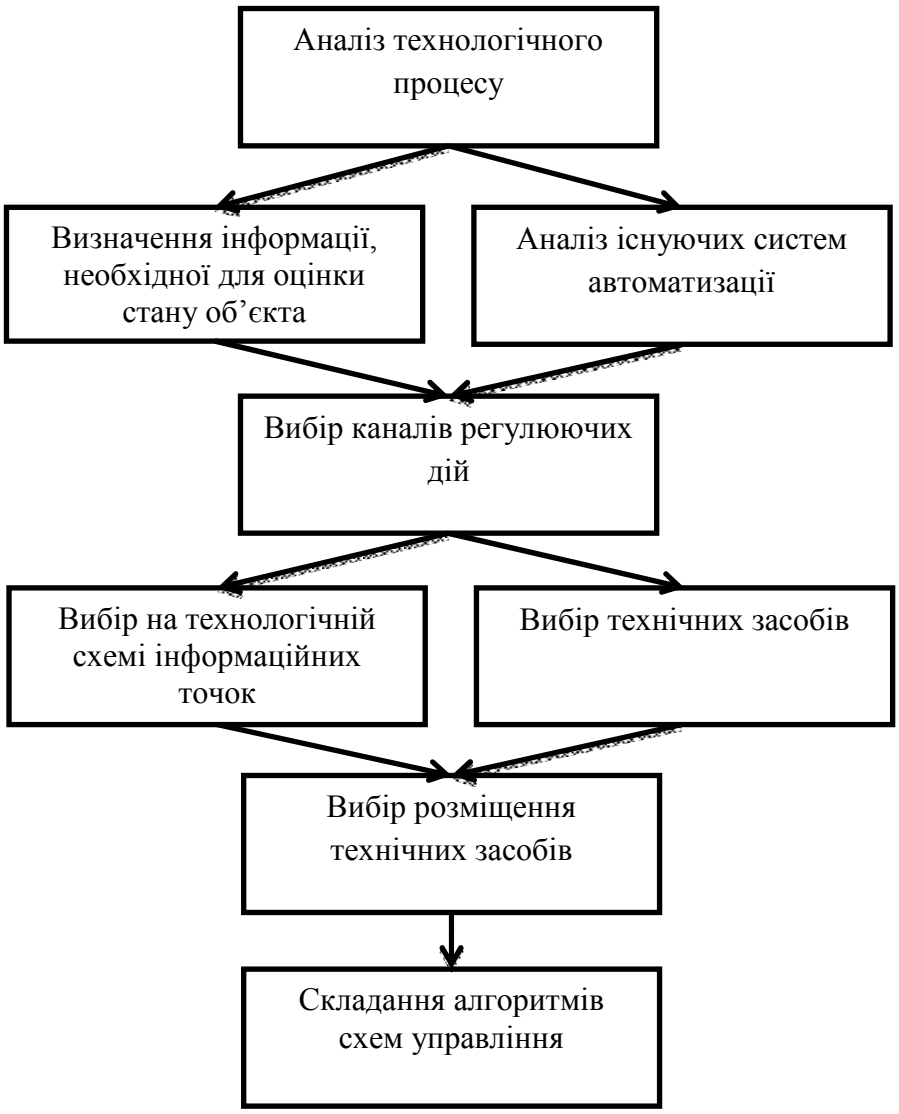


Рисунок 2.1 Структурно-логічна схема процедури розробки АС

Створення лабораторного комплексу для дисципліни «Проектування систем автоматизації» дозволить забезпечити наочне навчання та практику для студентів

спеціальності «Автоматизація та комп’ютерно інтегровані технології». Вони зможуть на прикладі імітаційної моделі працювати з різноманітними об’єктами, виконуючи на практиці всі етапи розробки автоматизованих систем.

На етапі аналізу технологічного процесу, студенти навчаться вибирати тех. процес, розбивати його на частини та отримувати вихідні дані. Наступний етап, визначення інформації, необхідної для оцінки стану об’єкта, дозволить навчити студентів правильно вибирати величини, інформація про які необхідна для оцінки стану об’єкта. Під час аналізу існуючих систем автоматизації, студенти навчаться здійснювати пошук необхідної технічної інформації використовуючи інтернет або навчальні посібники. При виборі каналів регулюючих дій, студенти навчаться аналізувати статичні і динамічні характеристики можливих каналів регулювальних дій та вибирати ті з них, що найкраще реалізують мету керування та відповідають технологічному регламенту. Наступний етап, вибір на технологічній схемі інформаційних точок, дозволить навчити студентів правильно вибирати точки для зняття інформації. Вибір типу мікропроцесорного контролера (МПК), розрахунок апаратного складу МПК, вибір засобів отримання інформації, вибір засобів подання інформації, вибір засобів реалізації регулювальних дій – все це студенти будуть робити на етапі вибору технічних засобів. На передостанньому етапі, студенти будуть розміщувати технічні засоби автоматизації на структурних схемах. На останньому етапі, студенти навчаться складати алгоритми роботи схем управління з урахуванням визначених функцій СА.

3 ТЕХНОЛОГІЧНА ЧАСТИНА

3.1 Робота у середовищі FACTORY I/O [14]

3.1.1 Панелі, кнопки та індикатори

Після запуску програми «FACTORY I/O» з'являється вікно (рис. 3.1), у якому є 3D віртуальний простір та інтерфейс користувача.

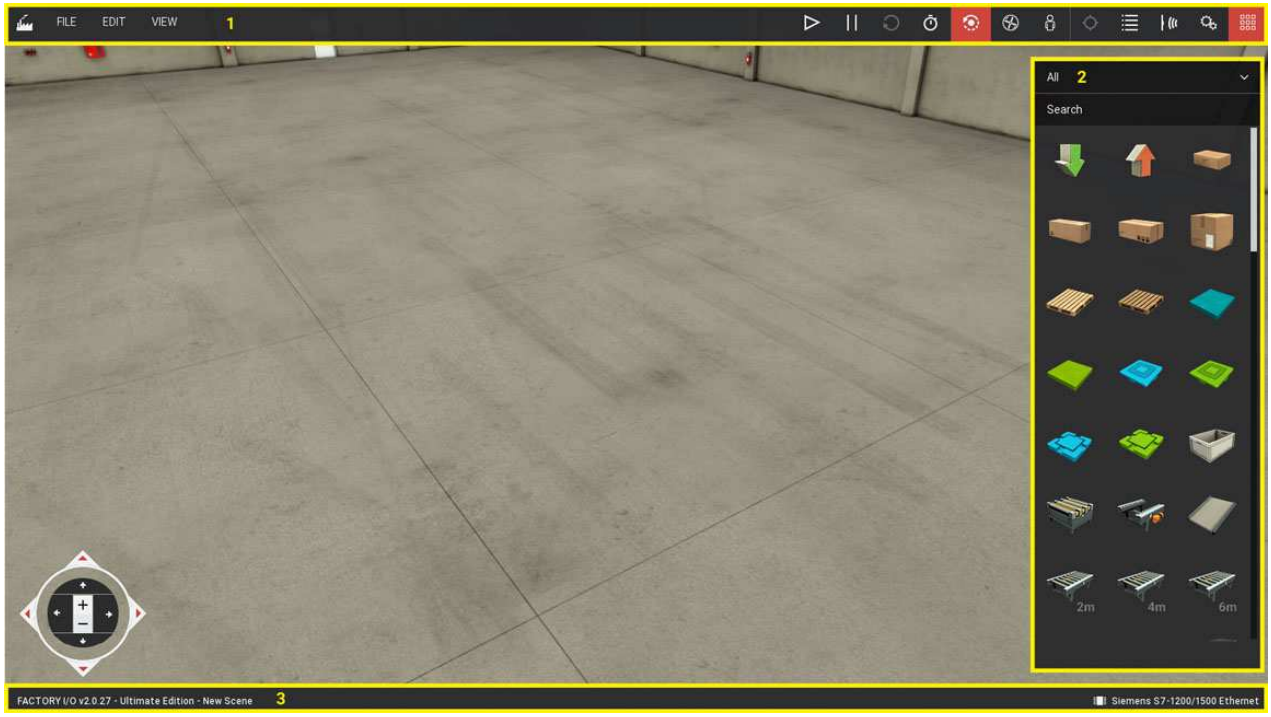


Рисунок 3.1 Вікно програми «FACTORY I/O»

Інтерфейс користувача складається з трьох частин:

1. Панель інструментів
2. Палітра
3. Рядок стану

У 3D-просторі віртуального цеху можна створювати нові чи використовувати готові моделі конвеєрів, автоматичних ліній та іншого устаткування для автоматизації виробництва. Дається змога комплектувати

автоматизовану систему різноманітними об’єктами з бібліотеки (палітри) та налаштовувати її під вимоги та потреби конкретної виробничої задачі.

Усі кнопки, які є на панелі інструментів, можна переглянути у таблиці 3.1

Таблиця 3.1 – Кнопки панелі інструментів

		**Welcome Menu**	Забезпечує швидкий доступ до документації, підручників, сцен та оновлень
		**Run/Edit**	Перемикач режиму симуляції (Запуск/Редагування)
		**Pause**	Зупинка симуляції
		**Reset**	Перезавантаження симуляції
		**Slow Motion**	Сповільнення часу в режимі симуляції (x10)
		**Orbit Camera**	Вибір режиму «Орбітальна камера»
		**Fly Camera**	Вибір режиму «Літаюча камера» для довільного вибору ракурсу
		**First Person Camera**	Вибір режиму «Камера від першої особи»
		**Follow a Part**	Дозволяє обрати частину для наслідування
		**Cameras Window**	Відкриття вікна камери
		**Sensors Tags**	Показує / Приховує теги датчиків
		**Actuators Tags**	Показує / Приховує теги виконавчих пристроїв
		**Palette**	Показує / Приховує палітру

У вікні панелі «Палітра» (рис. 3.2) відображаються всі компоненти, що є наявні у «FACTORY I/O». При створенні сцени, слід з використанням комп'ютерної миші перетягувати компоненти з палітри у віртуальний простір.

Шляхом вибору категорії з випадного списку, є можливість бачити лише ті компоненти, які належать до обраної категорії. Також дається можливість знаходити потрібні компоненти за допомогою уведення назви у полі пошуку Search.

Рисунок 3.2 Вікно «Палітра» у програмі «FACTORY I/O»

Панель «Рядок стану» (рис. 3.3) відображає інформацію про поточний стан «FACTORY I/O», а саме: поточна версія, випуск, назва сцени та обраний драйвер.

Рисунок 3.3 Рядок стану у програмі «FACTORY I/O»

Індикатор низької продуктивності (рис. 3.4) з'являється, коли «FACTORY I/O» не може оновити картинку моделювання швидше ніж 15 кадрів за секунду.

Це може призвести до небажаних візуальних стрибків та помилок у фізичних розрахунках.

Для покращення продуктивності можна виконати такі дії:

1) Зменшити розміри екрану (Option > Video);
2) Знизити якість відтворення відео (Option > Video);
3) Відключити V-Sync (Option > Video);
4) Зменшити кількість деталей у сцені, видаливши частини, які не використовуються.

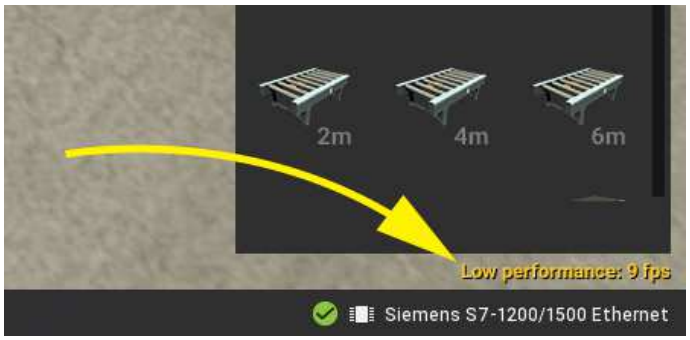


Рисунок 3.4 Індикатор низької продуктивності

3.1.2 Огляд палітри блоків у «FACTORY I/O»

3.1.2.1 Items

Основними елементами у категорії Items (рис. 3.5) є палети, коробки і так звана сировина з кришками. Детальніше про кожен елемент можна переглянути у таблиці 3.2.

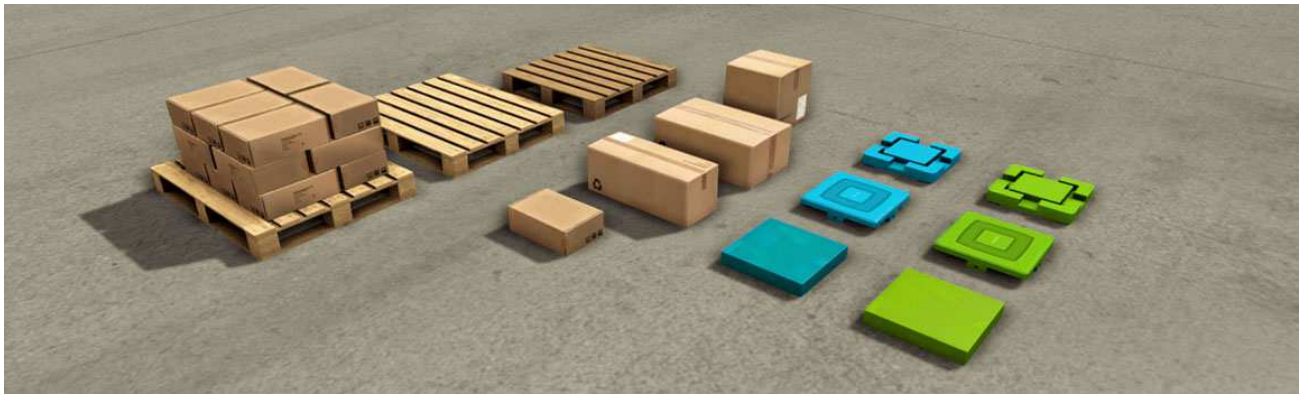


Рисунок 3.5 Елементи категорії Items

Таблиця 3.2 – Елементи категорії Items

Назва елемента	Характеристики (вага)	Опис елемента
Palletizing Box (Коробка для палетування) 	3 кг	Стандартна коробка для розміщення на палеті
Box (Коробка) 	S – 8 кг M – 10 кг L – 15 кг	Три різних типи коробок, кожна має різні розміри та вагу
Pallet (Піддон) 	20 кг	Дерев'яні піддони використовуються для укладання та транспортування всіх видів вантажу.
Square Pallet (Квадратний піддон) 	16 кг	
Raw material (Сировина) 	Blue – 8 кг Green – 8 кг	Пластмасова сировина, що використовується для виготовлення кришок і основ для базових виробів. Доступна у двох кольорах.
Product Lid (Кришка)	Blue – 5 кг Green – 5 кг	Частина виконана з пластику, яку можна

		зібрати з основою, щоб виготовити кінцевий продукт. Доступна у двох кольорах.
Product Base (Основа) 	Blue – 7 кг Green – 7 кг	Частина з пластику, яка може бути зібрана з кришкою для отримання кінцевого продукту. Доступна у двох кольорах.
Stackable Box (Ящик для складання) 	15 кг	Ящики для складання використовуються для транспортування в них таких предметів як сировина, кришка, основа)

3.1.2.2 Heavy Load Parts

Основними елементами у даній категорії є модулі придатні для перевезення важких вантажів (рис. 3.6). Детальніше про кожен модуль можна переглянути у таблиці 3.3.

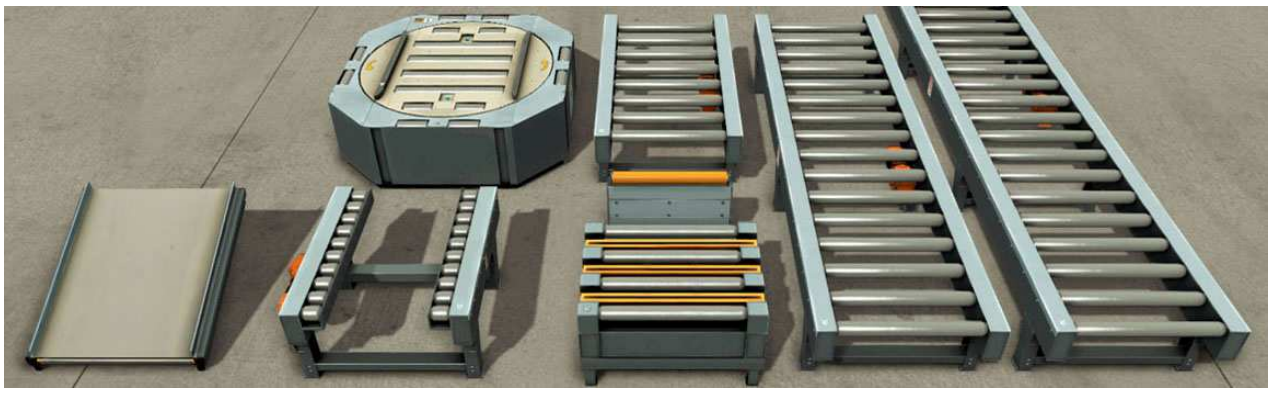
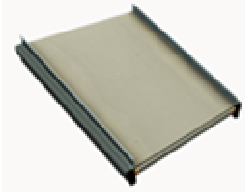


Рисунок 3.3 Елементи категорії Heavy Load Parts

Таблиця 3.3 – Елементи категорії Heavy Load Parts

Назва елемента	Характеристики	Опис елемента
Chain Transfer (Транспортер ланцюгового типу) 	Шлях до ланцюга: 0,04 м Макс. Транспортна швидкість: 0,45 м / с Макс. Швидкість ланцюга: 0,45 м / с	Використовується для перевезення вантажу на суміжні конвеєри. Він складається з завантажувальних роликів та трьох ланцюгових доріжок.
Loading Conveyor (Завантажувальний конвеєр) 	Радіус ролика: 0,046 м Макс. швидкість передачі: • 0,45 м/с (цифровий); • 0,8 м/с (аналоговий).	В основному використовується для завантаження / вивантаження вантажу на автонавантажувач. Може управлятися цифровими або аналоговими значеннями.
Chute Conveyor Low (Низький скат) 		Прямий полозний конвеєр, який зазвичай використовується для відправлення палет.
Roller Conveyor (Роликовий конвеєр) 	Ролик Радіус: 0,046 м; Доступні довжини: 2,4,6 м; Макс. швидкість передачі: • 0,45 м/с (цифровий); • 0,8 м/с (аналоговий).	Важільний роликовий конвеєр, можна керувати цифровими та аналоговими значеннями відповідно до обраної конфігурації.
Roller Stop (Блокувальний ролик)	Стан по замовчуванню: опущений; Висування: 0.1 м	Цей пневматично привідний пристрій може використовуватися для зупинки, накопичення

		або запобігання зіткнення матеріалів на конвеєрах.
Turnatable (Поворотний механізм) 	Радіус рулону: 0,045 м Макс. Транспортна швидкість: 0,45 м / с Таблиця швидкості повороту: 0,7 рад / с Діапазон ємнісних датчиків: 0 - 0,1 м	Використовується для сортування піддонів, обладнаний вільними обертальними периферійними роликами та попередньо встановл. датчиками.

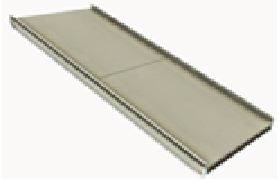
3.1.2.3 Light Load Parts

Основними елементами у даній категорії є модулі придатні для перевезення легких вантажів (рис. 3.7). Детальніше про кожен модуль можна переглянути у таблиці 3.4.

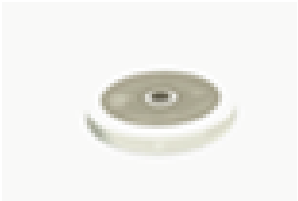
Рисунок 3.7 Елементи категорії Light Load Parts

Таблиця 3.4 – Елементи категорії Light Load Parts

Назва елемента	Характеристики	Опис елемента
Belt Conveyor (Стрічковий конвеєр)	Доступні довжини: 2,4,6 м; Макс. швидкість передачі: • 0,6 м/с (цифровий); • 3 м/с (аналоговий).	Використовується для транспортування легких вантажів. Може управлятися цифровими або аналоговими

		значеннями.
Belt Conveyor Gate (Стрічкові конвеєрні ворота) 	Можливий кут відкривання: 100 Макс. швидкість передачі: • 0,6 м/с (цифровий); • 3 м/с (аналоговий).	Призначені для забезпечення проходу для персоналу. Можуть управлятися цифровими або аналоговими значеннями.
Chute Conveyor (Високий скат) 		Прямий плунжерний конвеєр, в основному використовується для відправки предметів з ленточних конвеєрів.
Conveyor Scale (Конвеєрна вага) 	Макс. Швидкість передачі: 0,6 м/с; Конфігурації: 20 і 100 кг	Швидкісний конвеєр, що використовується для контролю ваги. Він вимірює різні діапазони ваги відповідно до обраної конфігурації.
Pivot Arm Sorter (Сортувальна рукоятка) 	Швидкість ременя: 2 м / с Ручна кутова швидкість: 5 рад/с Макс. кут повороту: 45 °	Лінійний ручний сортувальник, який оснащений редуктором. Оснащений ременем, який допомагає відхиляти предмети. Рука може обертати вліво або вправо відповідно до обраної конфігурації.
Pop Up Wheel Sorter (Роликовий сортувальник) 	Радіус коліс: 0,05 м Передавальна швидкість: 2,5 м / с Макс. кут повороту: 45 °	Використовується для переміщення предметів у три різні напрямки через поворотні ролики.

Pusher (Штовхач) 	Швидкість за замовчуванням: 1 м/с Макс. швидкість: 4 м/с Висування: 0,9 м	Пневматичний штовхач-сортувальник оснащений двома датчиками, що вказують на передні та задні обмеження. Може керуватись цифровими або аналоговими значеннями.
Stop Blade (Блокувальний механізм) 	Стан по замовчуванню: опущений; Висування: 0.12 м	Пневматичний пристрій, який використовується для зупинки або накопичення матеріалу.
Straight Spur Conveyor (Стрічковий трикутний конвеєр) 	Макс. передавальна швидкість: • 0,8 м/с (цифровий); • 3 м/с (аналоговий)	Використовується для точного злиття двох перпендикулярно встановлених конвеєрів. Може керуватись цифровими або аналоговими значеннями.
Positioner (Розміщувач) 	Доступний у двох конфігураціях: лівий, правий; Вертикальний хід: 0.373 м Хід затискача: 0.48 м	Пристрій, який використовується для послідовного розташування елементів у тому ж самому місці, затискаючи їх.
Aligner (Вирівнювач) 		Тонкий металевий каркас, який кріпиться до конвеєра, щоб запобігати падінню деталей при транспортуванні на великих швидкостях. Доступні чотири типи вирівнюючих пристроїв

		та декілька кольорів
Wheel Aligner (Колісний вирівнювач) 		Те ж саме, тільки у вигляді колеса
Metal Corner (Металевий кутник) 		Кронштейн являє собою металеву конструкцію, що використовується як бар'єр висоти і може використовуватися для приєднання датчиків. Доступний у декількох кольорах.
Bracket (Кронштейн) 		Металевий кут можна використовувати для кількох цілей, включаючи розміщення датчиків. Доступний у декількох кольорах.

3.1.2.4 Sensors

Основними елементами у даній категорії є датчики (рис. 3.8). Детальніше про кожен давач можна переглянути у таблиці 3.5.

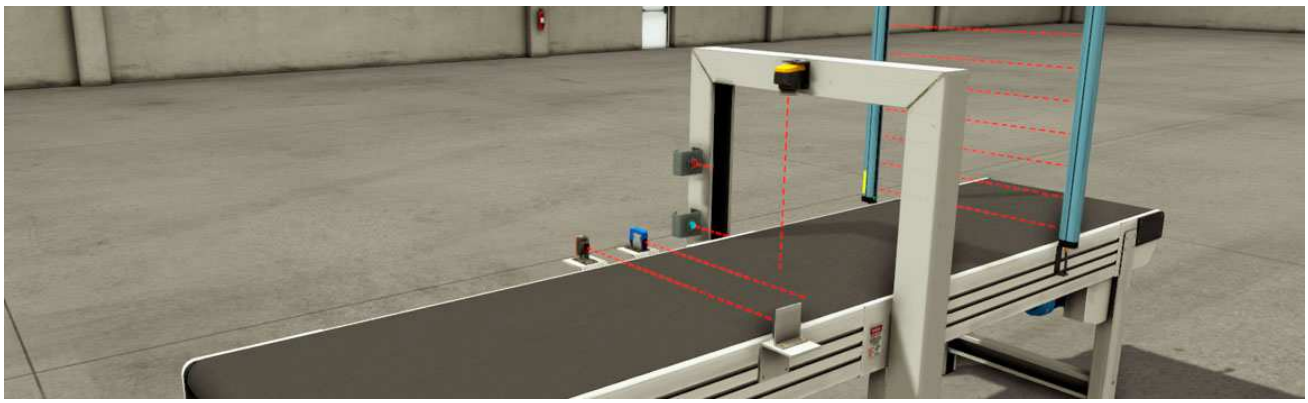


Рисунок 3.8 Елементи категорії Sensors

Таблиця 3.5 – Елементи категорії Sensors

Назва елемента	Характеристики	Опис елемента
Capacitive Sensor (Ємнісний датчик) 	LED: зелений (виявлення) Матеріали на які реагує: тверді речовини та рідини. Діапазон спрацювання: 0 - 0,2 м	Датчик близькості, який використовується для точного виявлення будь-якого матеріалу. Вихідне значення може бути цифровим або аналоговим.
Diffuse Sensor (Дифузний датчик) 	LED: червоний (виявлення) Матеріали на які реагує: тверді речовини. Діапазон спрацювання: 0 – 1,6 м	Дифузійний фотоелектричний датчик, який може виявити будь-який твердий об'єкт.
Inductive Sensor (Індуктивний датчик)	LED: червоний (виявлення) Матеріали на які реагує: Провідні матеріали. Діапазон спрацювання: 0 - 0,1 м	Використовується для тісного виявлення провідних матеріалів. Оснащений світлодіодним індикатором, що вказує на наявність об'єкта в

		межах його діапазону. Вихідне значення може бути цифровим або аналоговим відповідно до обраної конфігурації.
Light Array (Emitter and Receiver) (Випромінювач та приймач) 	Кількість випромінювачів: 8 штук. Матеріали на які реагує: Тверді матеріали Діапазон спрацювання: 1.5 м	Набір паралельних світлових пучків, що використовуються для створення сенсорного поля. Щоб забезпечити синхронізацію обох пристроїв , вони повинні стояти один навпроти іншого.
Vision Sensor (Датчик зору) 	LED: червоний (виявлення) Матеріали на які реагує: сировина, кришка, основа Діапазон спрацювання: 0.375 - 2 м	Датчик зору визначає сировину , кришки, основи продуктів та їхні відповідні кольори.
Retroreflective Sensor and Reflector (Світловідбиваючий датчик та рефлектор)  	LED: зелений (вирівнювання з рефлектором) LED: жовтий (промінь не переривається) Матеріали на які реагує: Тверді матеріали Діапазон спрацювання: 0 - 6 м	На відміну від інших датчиків, світло відбиваючий датчик потребує рефлектора. Для правильної роботи він повинен бути вирівняний з рефлектором. Він оснащений двома світлодіодними індикаторами, які вказують правильне вирівнювання (зелений) та стан виявлення

		(жовтий).

3.1.2.5 Operators

Основними елементами у даній категорії є кнопки, дисплеї та перемикачі (рис. 3.9). Детальніше про кожен елемент можна переглянути у таблиці 3.6.

Рисунок 3.9 Елементи категорії Operators

Таблиця 3.6 – Елементи категорії Operators

Назва елемента	Характеристики	Опис елемента
Emergency Stop (Аварійна зупинка) 	Тип кнопки: нормально замкнутого типу.	Двопозиційна червона тригерна кнопка. Зазвичай використовується в надзвичайних подіях.
Push Buttons (Кнопки)	Тип кнопок: нормально замкнутого типу.	Освітлена кнопка доступна у трьох різних

		кольорах (зелений: початок, жовтий: скидання та червоний: зупинка).
Light Indicators (Світлові індикатори) 	Доступні у чотирьох кольорах (синій, червоний, зелений та жовтий)	Індикатор підсвічування панелі. Зазвичай використовується на панелях інструментів для програм безпеки або індикації стану.
Selector (Перемикач) 	Стан по замовчуванню: 0	Неосвітлюваний селекторний перемикач використовується переважно для визначення поточного стану змінної або завдання.
Potentiometer (Потенціометр) 		Використовується для генерації аналогового значення пропорційного до обертання ручки. Вибрана конфігурація може визначати діапазон значень.
Digital Display (Цифровий дисплей)		Дозволяє відображати числові значення (плаваюче та ціле число) під час моделювання.

		Вибрана конфігурація може визначати діапазон значень.
Electric Switchboard (Електричний щиток) 		Використовується для проектування електричної дошки з операторами.
Column (Колона) 		Металеву конструкцію зазвичай використовують для утримання електричного щитка

3.1.2.6 Stations

Основними елементами у даній категорії є різноманітні великогабаритні готові станції (рис. 3.10). Детальніше про кожену станцію можна переглянути у таблиці 3.7.

Рисунок 3.10 Елементи категорії Stations

Таблиця 3.7 – Елементи категорії Stations

Назва елемента	Характеристики	Опис елемента
Machining Center (Центр обробки) 	Лампочки розміщені на верхній частині центру, дають інформацію про поточний стан. Зелена: центр вільний Жовта: центр зайнятий Червона: центр має помилку (неправильний елемент виявлений у вхідному відсіку)	Використовується для виготовлення кришок і основ з сировини. Коли виявляється новий матеріал у вхідному відсіку, він заванта-жується в машину з ЧПУ, яка розпочне виготовлення елемента. Кожен тип елемента вимагає іншого часу (кришки: 6 секунд , бази: 3 секунди), і після завершення операції робот розміщує елемент у відсіку для виходу.
Elevator (Ліфт)	Радіус роликів: 45 мм; Довжина платформи: 7 м; Швидкістьруху платформи: 0,68 м/с;	Використовується для перевезення всіх видів вантажів між поверхами. Ліфт оснащений двома світло відбиваючими датчиками, розміще-ними на кожному кінці

	Макс. швидкість: • 0,45 м/с (цифровий); • 0,8 м/с (аналоговий);	платформи. Ліфтом можна керувати цифро-вими або аналоговими значеннями.
Pick & Place (Станція для переміщення) 	Довжина по осі Y: 1.25 м; Довжина по осі X: 2.125 м; Довжина по осі Z: 0.5 м; Крок: 0.125 м Швидкість маніпулятора: 1.5 м/с; Кутова швидкість: 4.6 рад/с;	Використовується для переміщення легких вантажів (наприклад, картонні коробки) на інші конвеєри або палети. Має чотири ступені вільності. Керується цифровими або аналоговими знач.
Stacker Crane and Rack (Штеблерний кран та стійка)  	Довжина між вилками: 1.2 м; Довжина візка: 10.5 м; Довжина платформи: 6.625 м; Швидкість візка: 1.4 м/с; Швидкість висування вилок: 0.5 м/с; Швидкість платформи: 1.7 м/с; Кількість відсіків у стійці: 18;	Використовується для переміщення важких вантажів. Включає в себе візок, вертикальну платформу та дві вилки, які можуть ковзати в обидві сторони. Два лазерних датчики, які розміщені на візку та платформі, вимірюють горизонтальне та вертикальне положення платформи. Можна керувати цифровими, числовими та аналоговими значеннями відповідно до конфігурації.
Palletizer (Палетизатор)	Довжина штовхача: 0.88 м; Довжина ліфта: 1.75 м;	Використовується для складання картонних коробок (Palletizing Box) на піддони.

	Швидкість ліфта: 2 м/с;	
Two-Axis Pick & Place (Двох осьова станція для переміщення) 	Довжина по осі X: 1.125 м; Довжина по осі Z: 0.625 м; Швидкість руху захвату: 2 м/с;	Використовується для збирання кришок з основами або для вибору та розміщення предметів з одного місця в інше.
Tank (Бак з рідиною) 	Висота: 3 м; Діаметр: 2 м; Радіус випускної труби: 0,125 м; Макс. вхідний потік: 0,25 м³/с; Макс. вихідний потік: 0,3543 м³/с;	Резервуар з двома регулюючими клапанами та ємнісним датчик рівня, який може використовуватися для регулювання потоку рідини.

3.1.2.7 Warning Device

Основними елементами у даній категорії є попереджувальні пристрої (рис.3.11). Детальніше про кожен пристрій можна переглянути у таблиці 3.8.

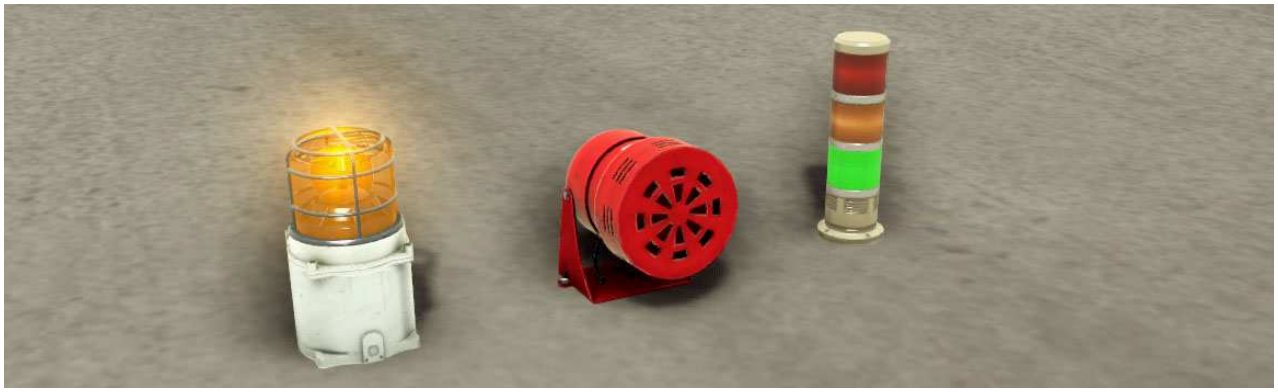


Рисунок 3.11 Елементи категорії Warning Device

Таблиця 3.8 – Елементи категорії Warning Device

Назва елемента	Характеристики	Опис елемента
Alarm Siren (Сирена тривоги) 		Подає сигнал тривоги під час аварійних ситуацій.
Stack Light (Циліндричний ліхтар) 		Використовується як візуальний індикатор станів і процесів обладнання. Виготовлений із трьох різних кольорів: червоного, жовтого та зеленого кольорів.
Warning Light (Попереджувальний ліхтар) 		Має обертаючий механізм та подає потужні візуальні сигнали попередження.

3.1.2.8 Walkways

Основними елементами у даній категорії є різноманітні пішохідні конструкції (рис.3.12). Детальніше про кожну конструкцію можна переглянути у таблиці 3.9.

Рисунок 3.12 Елементи категорії Walkways

Таблиця 3.9 – Елементи категорії Walkways

Назва елемента	Характеристики	Опис елемента
Handrails (Поручні) 	Мають такі модифікації: • S; • M; • L; • XL; • кутовий поручень; • сходовий поручень	Металеві трубчасті конструкції які використовуються для забезпечення безпеки операторам цеху. Доступні у кількох кольорах.
Platform (Платформа) 	Мають такі модифікації: • S; • M; • L; • XL; • Платформний стовп;	Металеві конструкції використовуються для створення підлоги на потрібній висоті. Доступні у кількох кольорах.
Safeguard (Огорожа) 	Мають такі модифікації: • S; • L; • I;	Металева огорожа, яка застосовується для від-городження периметра біля робочих зон. Доступні у кількох кольорах.

Stairs (Сходи) 		Металеві конструкції для забезпечення доступу до вищих рівнів. Можна компонувати разом з сходовими поручнями. Доступні у кількох кольорах.

3.1.3 Камери у «FACTORY I/O»

Навігація та управління відеокамерами є ключовим моментом у використанні «FACTORY I/O». Майже кожне завдання вимагатиме навичок правильного використання цього інструменту для досягнення ефективних результатів при роботі з програмою. Камери використовуються при навігації у 3D-просторі, при редагуванні і створенні сцен та роботі з компонентами тощо. Важливо у максимальній мірі оволодіти цим інструментом, бо від цього буде залежати зручність при роботі з програмою.

Існує три типи камер (рис. 3.5): Orbit (1), Fly (2) та First Person (3). Кожна камера краще підходить для певних завдань, і користувач сам обирає найефективнішу залежно від дій, що виконуються. Можна робити перемикання між камерами кнопками на панелі інструментів, як показано на рисунку 3.13.

Рисунок 3.13 Перемикання між типами камер у програмі «FACTORY I/O»

Режим «Орбітальної камери» (Orbit Camera) було розроблено для полегшення дій при редагуванні, отже, він стає найбільше придатним при створенні сцен. Ця камера працює шляхом обертання навколо точки зацікавлення, яка встановлюється за допомогою подвійного клацання лівою кнопкою миші (ЛКМ) (табл. 3.10) на будь-якій частині 3D-простору і позначається білою крапкою.

Крім того, кожен раз, коли частина переміщається, точка зацікавлення автоматично встановлюється в центрі деталі. Ви можете зберегти точку зацікавлення у початковому положенні, натиснувши і утримуючи LEFT SHIFT.

Після того, як точка зацікавлення визначена, можна обертати камеру навколо неї, утримуючи ПКМ та переміщаючи мишку.

Нові деталі, які додаються з палітри, створюються на висоті, яка визначається цією точкою, за винятком частин, які зазвичай розташовуються на підлозі, таких, як конвеєри, станції тощо.

Для того, щоб розмістити невелику компоненту на більшій, наприклад, датчик на конвеєрі, спочатку визначають точку зацікавлення, де потрібно помістити невелику компоненту і потім цю частину слід перетягнути з палітри.

Таблиця 3.10 – Комбінації клавіш при роботі з орбітальною камерою

Комбінація клавіш	**Дія**
x2 ЛКМ	Встановлює точку зацікавлення. Камера буде обертатися навколо цієї точки та компоненти будуть розташовуватися на цій висоті.
ПКМ + переміщення	Обертає камеру навколо точки зацікавлення
Середня кнопка миші (колесо) + переміщення	Встановлює камеру горизонтально
Колесо миші	Зближення та віддалення камери
Пробіл	Скидає положення і позицію камери на початкову за замовчуванням
W | Up ↑	Рух камери вперед
S | Down ↓	Рух камери назад

D | Right →	Рух камери вправо
A | Left ←	Рух камери вліво

Режим «Літаючої камери» використовується для вільного переміщення у 3D-просторі. Ця камера стикається з компонентами сцени, але не виявляється датчиками. Для полегшення роботи з цим режимом камери, можна скористатись гарячими клавішами, які зображені у таблиці 3.11.

Таблиця 3.11 – Комбінації клавіш при роботі з літаючою камерою

Комбінація клавіш	Дія
x2 ЛКМ	Переміщує камеру в місце, де було двічі натиснуто ЛКМ
ПКМ + переміщення	Обертання камери
Колесо миші	Переключає камеру у вертикальне положення
ЛКМ + ПКМ	Рух камери вперед
W | Up ↑	Рух камери вперед
S | Down ↓	Рух камери назад
D | Right →	Рух камери вправо
A | Left ←	Рух камери вліво

Режим «Камери від першої особи» встановлюється на висоті 1,80 м для моделювання і спостереження очима людини на фабриці. Ця камера стикається з частинами сцени, але не виявляється датчиками. Гарячі клавіші для цього режиму камери, зображені у таблиці 3.12.

Таблиця 3.12 – Комбінації клавіш при роботі з камерою від першої особи

Комбінація клавіш	Дія
x2 ЛКМ	Переміщує камеру, де було двічі натиснуто ЛКМ
ПКМ + переміщення	Обертання камери
ЛКМ + ПКМ	Рух камери вперед
W	Рух камери вперед
S	Рух камери назад
A	Рух камери вліво
D	Рух камери вправо
Пробіл	Стрибок

3.1.4 Режими роботи програми «FACTORY I/O»

«FACTORY I/O» може працювати у двох різних режимах: редагування та запуску. В режимі редагування (Edit Mode) є можливість відкривати, редагувати, створювати, зберігати та видаляти сцени. У самих сценах можна змінювати розміщення компонентів, трансформувати їх та модифікувати.

Компонента створюється шляхом перетягування потрібного елемента з палітри у 3D-простір. Також, передбачено можливість створювати нові компоненти дублюванням уже раніше створених: для цього слід обрати потрібну компоненту, натиснути й утримувати клавішу ALT та перетягнути скопійовану частину на нове місце у робочому просторі. Слід звернути увагу на те, що, якщо дубльована частина виділена червоним прямокутником (рис. 3.14), то вона перетинає якусь іншу готову компоненту і буде видалена при такому хибному встановленні

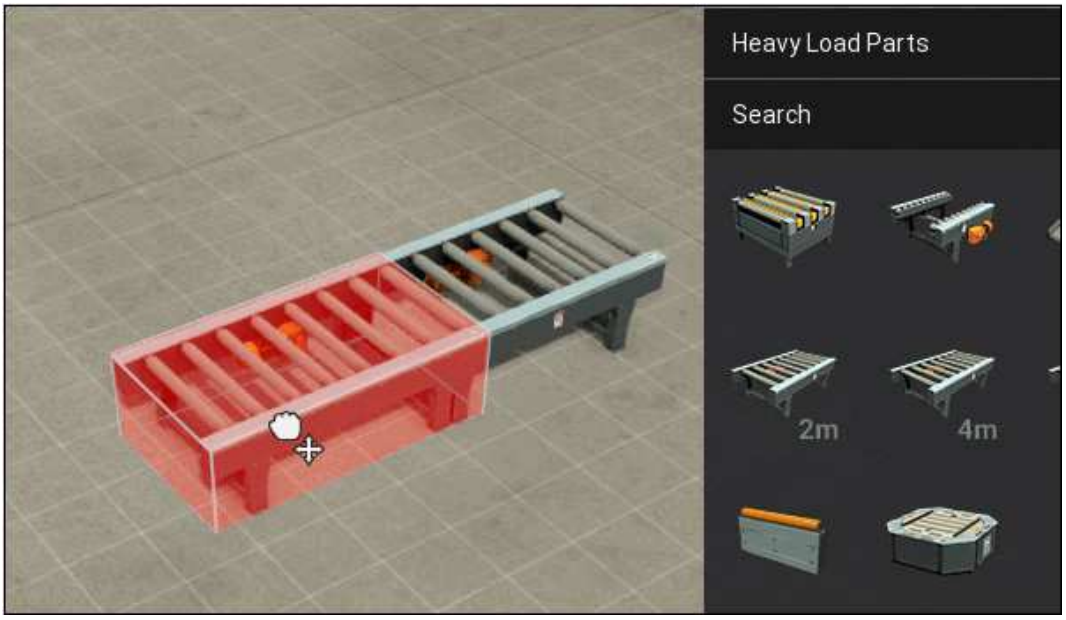


Рисунок 3.14 Перетинання дубльованою частиною іншої частини

Окрему частину можна виділити просто клацнувши на ній ЛКМ (рис. 3.15). Як альтернативу, також можна обрати декілька частин одразу за допомогою

Rectangle Selection Tool: затиснути ліву кнопку миші на тлі робочого простору сцени і намалювати прямокутник, який перетинатиме ті частини, які необхідні. Після цього виділені елементи будуть додані до вибірки виділення.

Можна додавати або видаляти частини з виділення, затиснувши Ctrl та натискаючи ЛКМ при обранні потрібних.

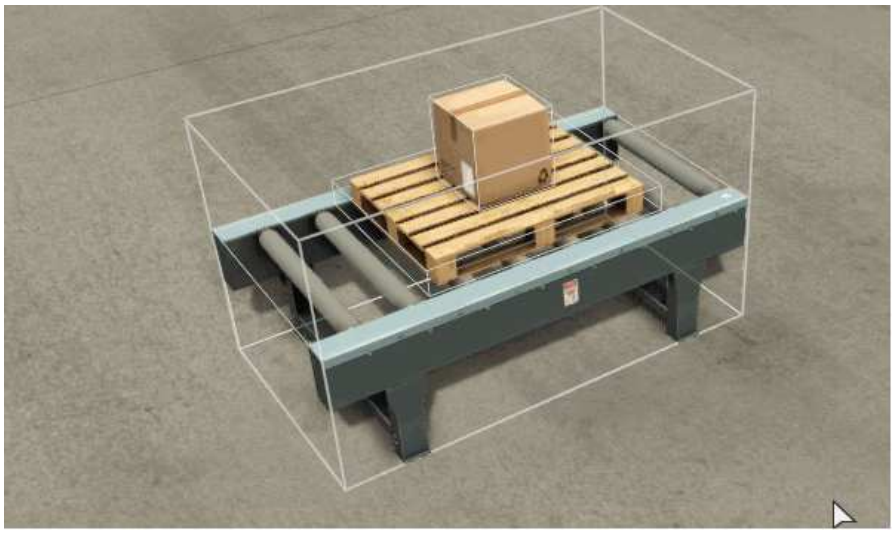


Рисунок 3.15 Виділення елемента

Щоб видалити частину або групу потрібно клацнути на ній або виділити потрібне і натиснути клавішу Delete.

Щоб перемістити потрібну частину або групу потрібно клацнути на ній або виділити і утримуючи ЛКМ перемістити у потрібне місце. За замовчування, частини переміщуються по горизонталі (рис. 3.16).

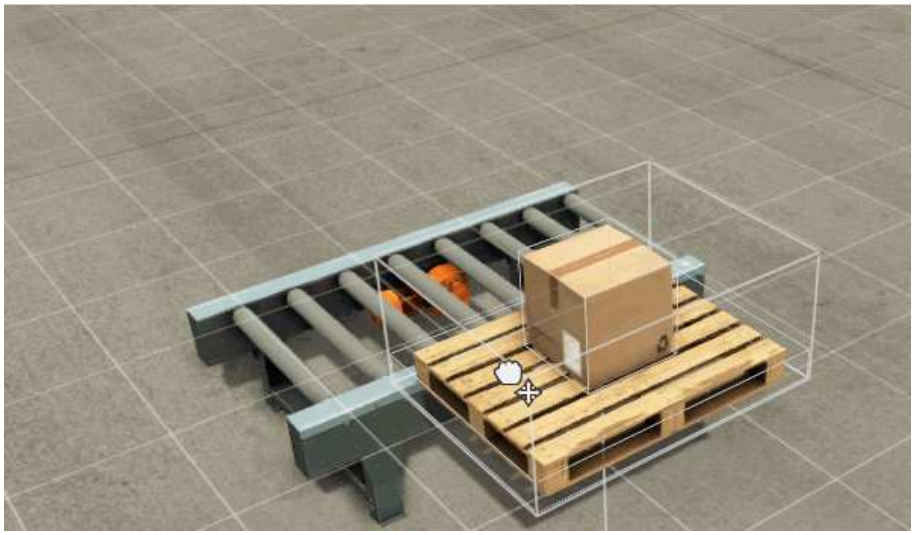


Рисунок 3.16 Переміщення елементів по горизонталі

Для того, щоб переміщувати елементи вертикально (рис. 3.17), потрібно додатково затиснути клавішу V і, утримуючи ЛКМ, перетягувати вгору або вниз.

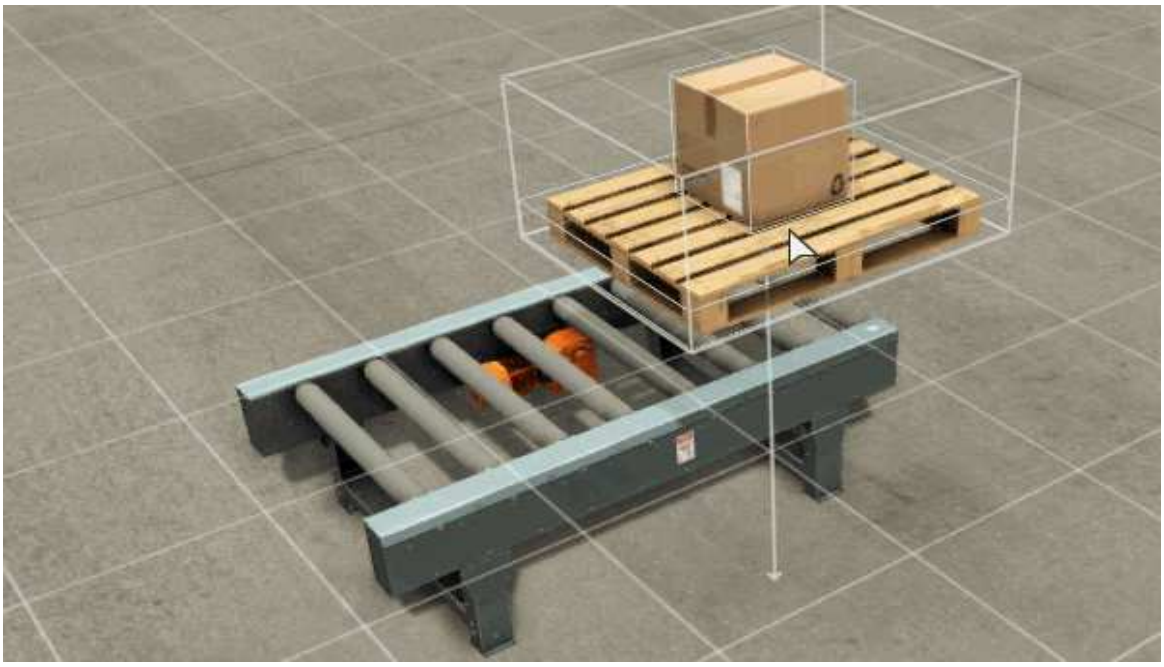


Рисунок 3.17 Переміщення елементів по вертикалі

«FACTORY I/O» використовує інтелектуальний алгоритм зіткнень, який дозволяє розміщувати частини лише на дійсних позиціях. Ця особливість симулятора робить створення 3D-сцени зручнішим та дозволяє робити проектування у більшій мірі схожим до реальності.

Щоб здійснити повертання, необіхдно обрати частину або групу і натиснути одну з таких клавіш: Y (Yaw); R (Roll); T (Pitch). Залежності від того, як потрібно обертати – використовується відповідна кнопку. Слід звернути увагу, що більшість елементів дозволяють повороти лише на 90˚, а датчики можуть вільно обертатися навколо власної осі.

Частини, які при обертанні обведені червоним прямокутником і перетинають інші частини або не поміщаються і не встановлюються в правильне положення – будуть видалені. Приклад обертання коробки розміщеної на палеті можна побачити на рисунку 3.18.

Рисунко 3.18 Обертання елементів

Частини можуть бути згрупованими для полегшення виконання декількох операцій одразу. Слід обрати елементи, які потрібно згрупувати та натиснути комбінацію клавіш Ctrl + G, щоб згрупувати їх разом. Можна обирати декілька окремих частин. Для цього потрібно обирати їх, затиснувши клавішу Ctrl. Для того, щоб розгрупувати групу – слід обрати цю групу і натиснути Ctrl + G повторно.

У робочому режимі сцена моделюється в режимі реального часу і може керуватися вручну або за допомогою зовнішнього контролера (наприклад PLC).

Курсор миші «рука» ідентифікує деталі, з якими можна взаємодіяти. Щоб перемістити віртуальний об’єкт слід клацнути ЛКМ і перетягти його інтерактивну частину. Щоб заблокувати обертання під час руху слід утримувати клавішу Shift.

Моделювання може бути припинене і відновлене в будь-який момент. Призупинення сцени можна зробити у будь-який момент часу, що дозволяє перевіряти стан кожного приводу і датчика, а також здійснити налагоджувальні операції на контролері.

Крім того, сцена може бути запущена в повільному русі (Slow Motion), що дозволяє ретельно аналізувати поведінку приводів, датчиків і елементів. Це може бути корисним, особливо на сценах зі швидкорухомими частинами і деталями.

3.1.5 Відображення стану датчиків і приводів

Кожен датчик або привід має один або кілька станів. Стани використовуються для зв'язування значень виконавчих механізмів та датчиків з контролером. Крім того, стани також можуть бути використані для управління виконавчими механізмами вручну та аналізу виконання програми.

Стан описується іменем та значенням. При створенні елементу імена автоматично призначаються станам. Зазвичай краще переназвати стани короткими і описовими іменами, тому що вони будуть використовуватися при відображенні виконавчих елементів та датчиків у віртуальному середовищі. Значення можуть бути трьох різних типів даних, залежно від типу та конфігурації датчиків та приводів: Boolean (логічне значення) для станів ввімкнення/вимкнення, Float для аналогових значень (дійсних чисел) та Integer для цілочисельних даних. Кожен тип даних має свій колір. Відповідно можна динамічно відслідковувати яка змінна якого типу даних у вікні поєднання тегів датчиків та виконавчих пристроїв (рис. 3.19).

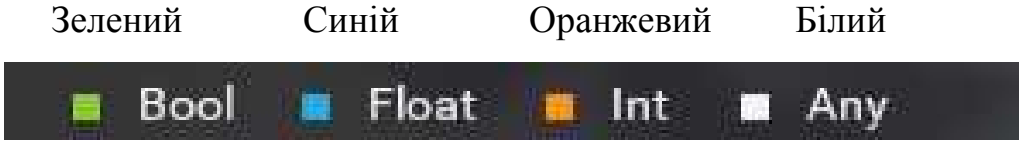


Рисунок 3.19 Типи змінних та їх кольори

Можна показувати або приховувати відображення станів датчиків (Sensors Tags) та виконавчих приводів (Actuators Tags) через натискання відповідних піктограм на панелі інструментів (рис. 3.20).

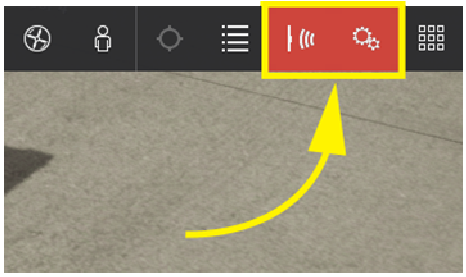


Рисунок 3.20 Кнопки включення/виключення відображення станів датчиків і виконавчих механізмів

Відображення станів можна прикріпити у верхньому лівому куті вікна, натиснувши на відповідних датчиках чи виконавчих механізмах лівою кнопкою миші. Ці стани будуть відображатись постійно, незалежно від положення камери огляду (рис. 3.21). Крім того, ці стани можна перейменовувати та перемикати вручну для виявлення помилок у програмі при відлагодженні. Щоб сховати всі прикріплені відображення станів, слід натиснути View > Clear Docked Tags.

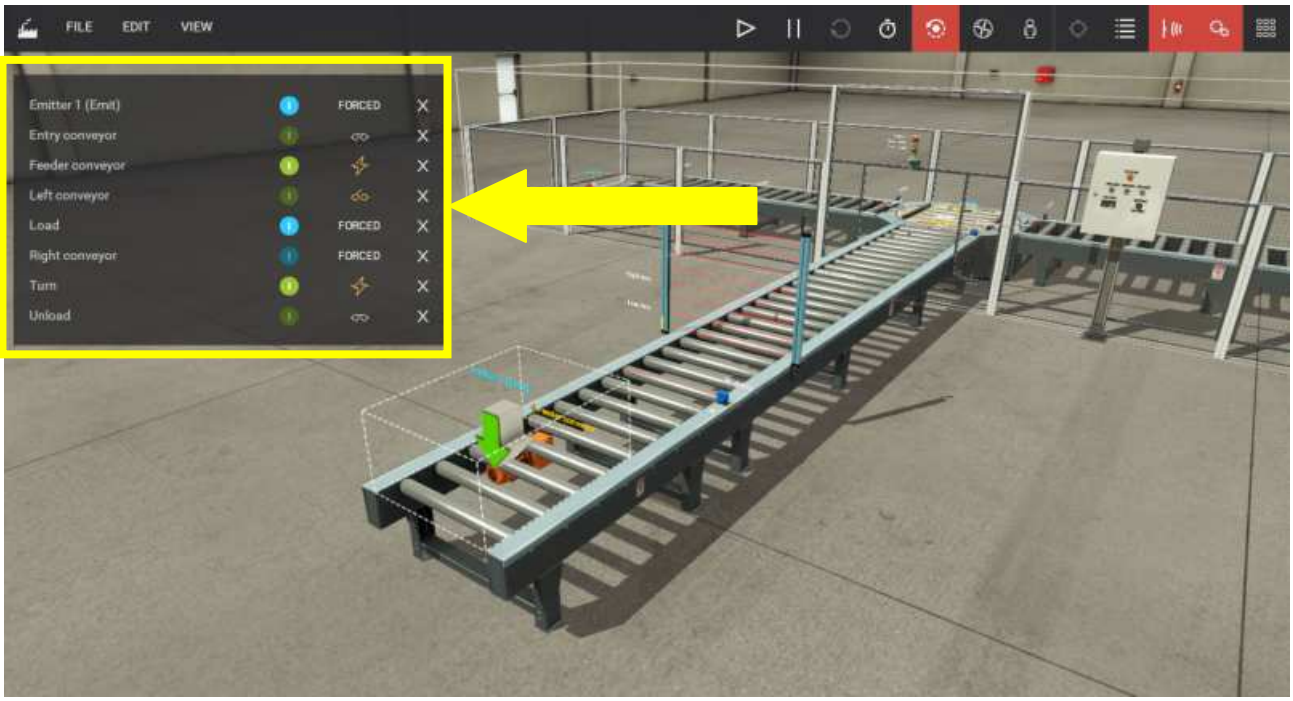


Рисунок 3.21 Відображення станів окремих компонентів сцени

3.1.6 Примусова зміна стану датчиків і приводів

Стани датчиків і виконавчих механізмів можуть бути примусово змінені натисканням ЛКМ (лівої клавіші мишки) на кнопці, слайдері або введенні у полі стану (залежно від типу даних) (табл. 3.13). При натисканні кнопки виконавчого механізму, можна перевизначити значення його стану, отримане з контролера. Примусово змінюючи стани виконавчих елементів, можна вручну виконувати роль контролера та керувати роботою обладнання. Натисканням на кнопки датчиків можна перезадати їх стан і відправити відповідний сигнал на контролер незалежно від стану обладнання.

Таблиця 3.13 – Зміни стану різних типів

Тип	Значення ON	Значення OFF	Зміна величини
Логічний (Bool)			ЛКМ для перемикання on/off.
Дійсне число (Float)			ЛКМ і пересування для встановлення потрібного значеня.
Цілочисельний (Integer)	4	**0**	ЛКМ і ввести нове значення**.**

Актуалізація примусової зміни стану здійснюється натисненням ЛКМ кнопки RELEASE. Стани компонентів сцени можуть бути в автоматичному (зелений кружечок) та в ручному режимі (синій кружечок) (рис. 3.22).

Рисунок 3.22 Стани компонентів сцени

3.1.7 Моделювання зміни станів

Кожна сцена включає в себе чотири вбудовані стани (рис. 3.23, табл. 3.14), які контролер може використати для отримання релевантних даних про моделювання (симуляцію).

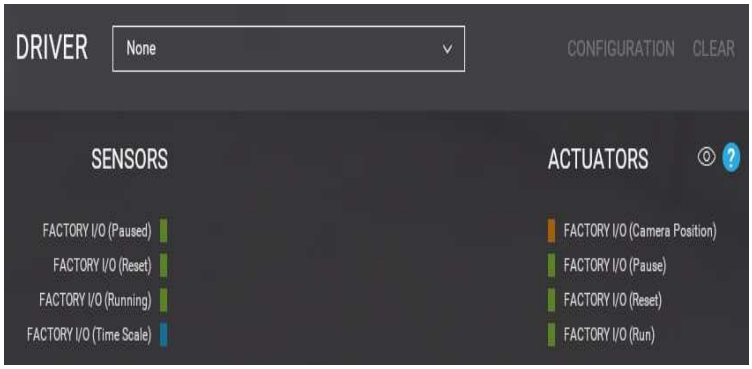


Рисунок 3.23 Вбудовані стани

Таблиця 3.14 – Стани симуляції FACTORY I/O

Стан симуляції	**Опис**
FACTORY I/O (Paused)	True, коли симулювання роботи є на паузі
FACTORY I/O (Reset)	True (протягом 1с), перезапуск симулювання
FACTORY I/O (Running)	Коли True, симулювання відбувається
FACTORY I/O (Time Scale)	Масштабування симуляції у часі: • Режим паузи або редагування = 0 • Сповільнене виконання = 0.1 • Виконання у реальному часі = 1

3.2 Робота у середовищі CONTROL I/O [15]

3.2.1 Загальні поняття та визначення SoftPLC CONTROL I/O

Soft PLC – це програмне середовище, яке використовується для моделювання та імітації роботи контролера на базі ПК. При використанні soft PLC, частина ресурсів ЦП резервується для моделювання PLC-системи управління, а інша частина виділяється для операційної системи. Робота soft PLC-контролера ідентична до роботи звичайного контролера: він реалізує логіку управління на стандартній мові програмування FBD за IEC 61131-3. Soft PLC-

контролер приймає дані від польових пристроїв, обробляє їх за допомогою логіки, реалізованої мовою програмування IEC 61131-3, та направляє вихідні дані польовим пристроям через HMI-інтерфейс.

CONTROL I/O - це простий у використанні, незалежний від фірми SoftPLC, розроблений спеціально для Factory I/O. CONTROL I/O забезпечує простий та інтуїтивно зрозумілий інтерфейс, навіть для початківця у цій галузі. У CONTROL I/O логіка роботи контролера задається за допомогою діаграми функціональних блоків (рис. 3.24).

Рисунок 3.24 Вікно CONTROL I/O

3.2.2 Огляд інтерфейсу користувача

Інтерфейс CONTROL I/O (рис. 3.25) складається з 7 секцій(табл. 3.15):

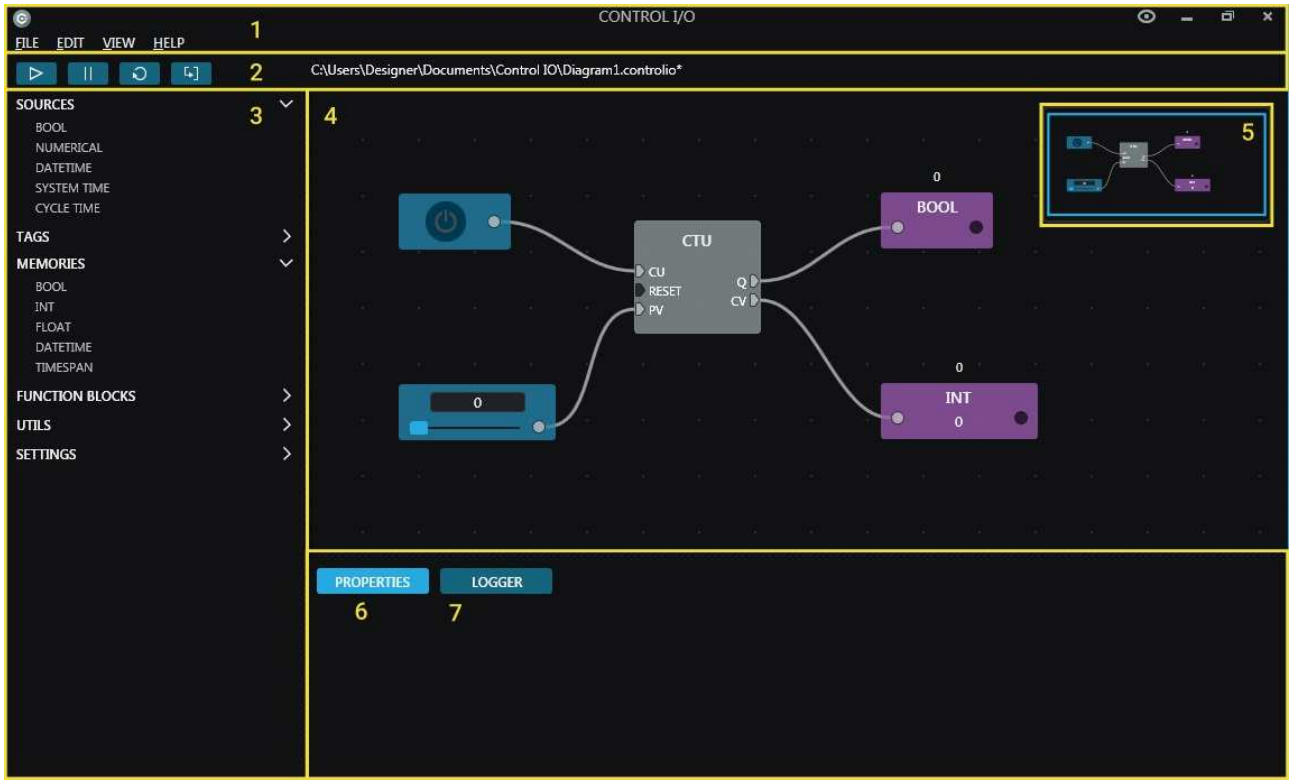


Рисунок 3.25 Інтерфейс CONTROL I/O

Таблиця 3.15 – Секції інтерфейсу CONTROL I/O

№	Секція	Опис
1	Меню файлу	Надає доступ до загальних команд.
2	Панель інструментів	Керує виконанням програми. Використовується для запуску, паузи, скидання або виконання одного кроку діаграми. За замовчуванням кнопки панелі інструментів також керують Factory I/O. Цю функцію можна відключити у налаштуваннях.
3	Палітра блоків	Список усіх блоків, згрупованих за типом. Тут також знаходиться меню налаштування програми.
4	Полотно	Місце, де малюється схема функціонального блоку.
5	Огляд полотна	Візуальне зображення усієї діаграми.
6	Властивості	Відображає властивості та відповідну інформацію для вибраних блоків.

7	Журнал	Відображає інформацію про виконання програми.

3.2.3 Огляд палітри блоків у CONTROL I/O

Усі блоки у палітрі представляють собою значення якогось певного типу (Boolean, Integer, Float тощо) або функцію між входами та виходами (Лічильник, Таймер тощо). У середовищі CONTROL I/O усі блоки створюються у вигляді «сокетів» (рис. 3.26), які використовуються для з'єднання різних блоків разом. З'єднуючи сокети разом, можна визначати, які дані будуть використані та у якій послідовності (зліва направо). Вихідні сокети можна пов'язувати лише з вхідними. Усю палітру блоків CONTROL I/O можна розділити на декілька категорій (табл. 3.16).

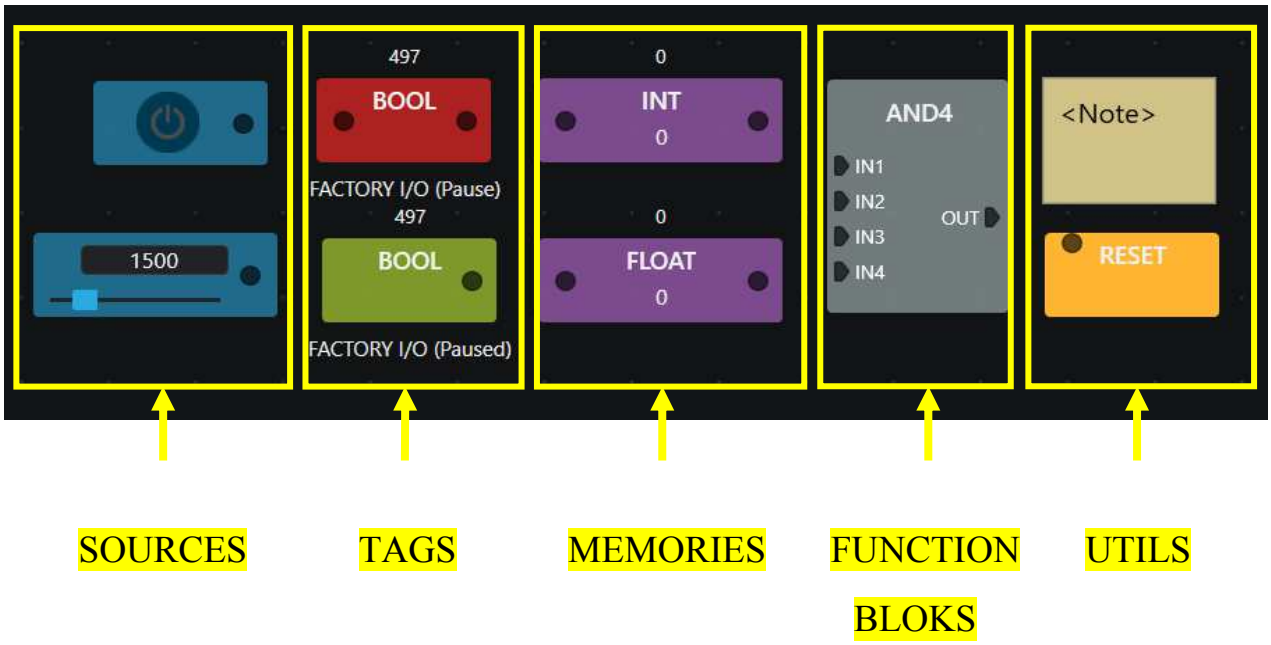


Рисунок 3.26 Сокети в CONTROL I/O

Таблиця 3.16 – Організація блоків за категоріями

Категорія блоків	**Опис**
Sources	Блоки, що використовуються для генерування даних.

(Джерела)	Можна знайти відповідний блок для кожного типу даних, що підтримується, а також блоки, що забезпечують системний час та час циклу.
Tags (Теги)	Точки вводу та виводу, які наявні у Factory I/O. Вони автоматично виявляються за допомогою CONTROL I/O.
Memories (Пам’ять)	Використовується для відслідковування значень змінних протягом виконання програми.
Function Blocks (Функціональні блоки)	Функціональні блоки, що описують функцію між входами та виходами. Можуть бути логічні, арифметичні, таймери, лічильники тощо.
Utils (Утиліти)	Має сокет Note (Примітка) та Reset (Скидання). Блок скидання може використовуватися як діаграма для скидання самого себе.

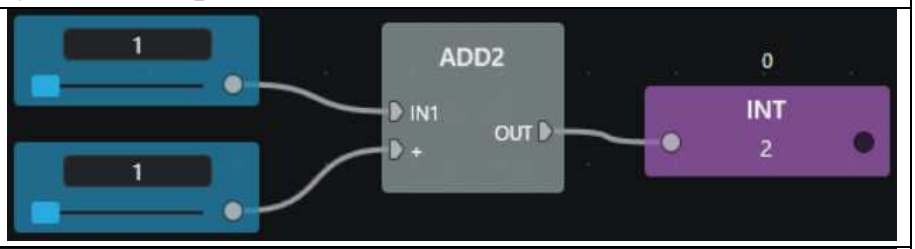
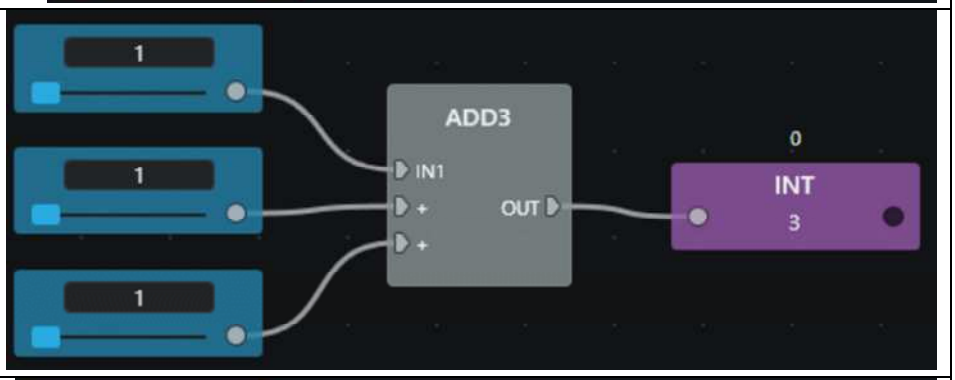
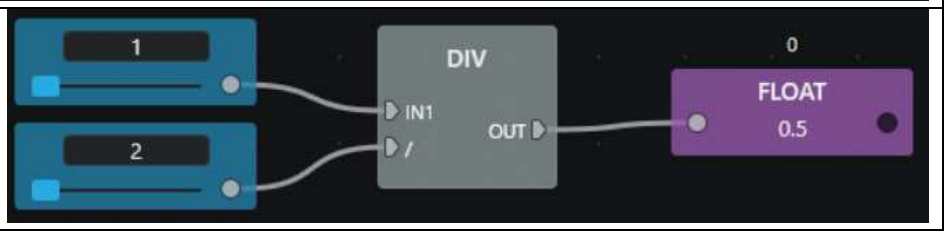
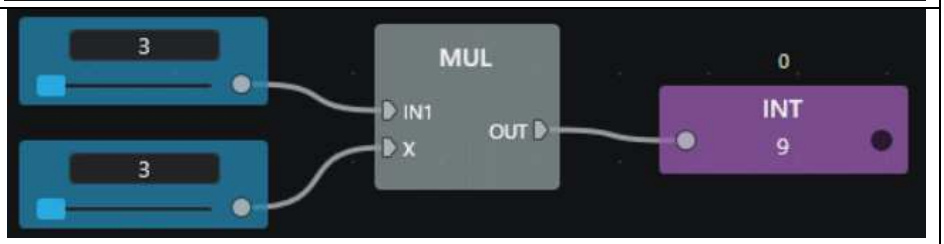
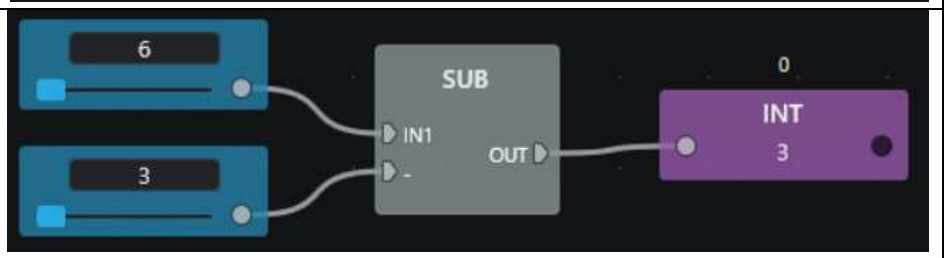
Блоки вводу (зелений), виводу (червоний) та пам'яті (фіолетовий) мають адресу та назву. Адреса визначає положення цього блоку у внутрішній пам'яті CONTROL I/O (подібно до реального PLC), тоді як ім'я допомагає ідентифікувати блок без необхідності відстежувати адресу. Ця пам’ять поділяється з FACTORY I/O, тобто блок з тією ж адресою, що і елемент сцени, має те саме значення. Використовуючи адреси, ви можете легко зіставити загальну діаграму на елементи сцен, встановивши правильні адреси для входів та виходів.

3.2.4 Огляд функціональних блоків

3.2.4.1 Arithmetic (Арифметика)

Це блоки (табл. 3.17), які виконують арифметичні операції. Для цих блоків можна використовувати будь-який числовий тип даних.

Таблиця 3.17 – Арифметичні функціональні блоки

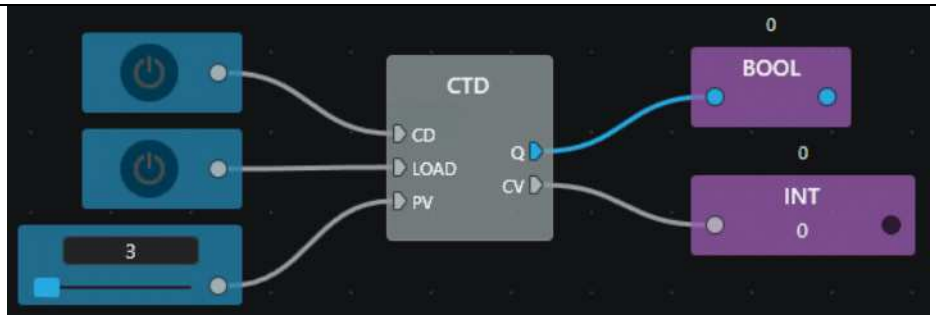
Опис	Візуальне зображення «сокета»
ADD2 Додає два числа.	
ADD3 Додає три числа.	
DIV Ділить одне число на інше.	
MUL Множить два числа.	
SUB Віднімає одне число від іншого.	

3.2.4.2 Counters (Лічильники) (табл. 3.18)

Таблиця 3.18 – Функціональні блоки – лічильники

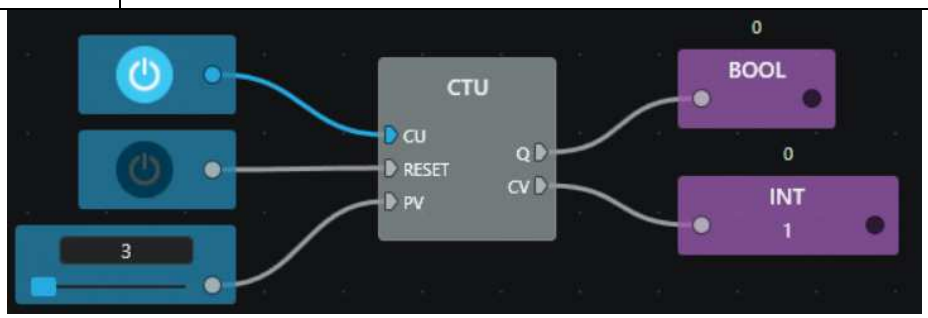
Опис	Візуальне зображення «сокета»

CTD (Таймер)
Робить відлік числа та встановлює вихід (Q) в True, коли досягне 0.



Входи	
CD	Запуск таймера
LOAD	Скидання таймера
PV	Початкове значення
Виходи	
Q	Q є TRUE, коли CV=0. У всіх інших випадках Q є FALSE.
CV	Поточне значення

CTU (Лічильник)
Підраховує числа, встановлюючи вихід (Q) в True, коли воно досягне заданого значення.



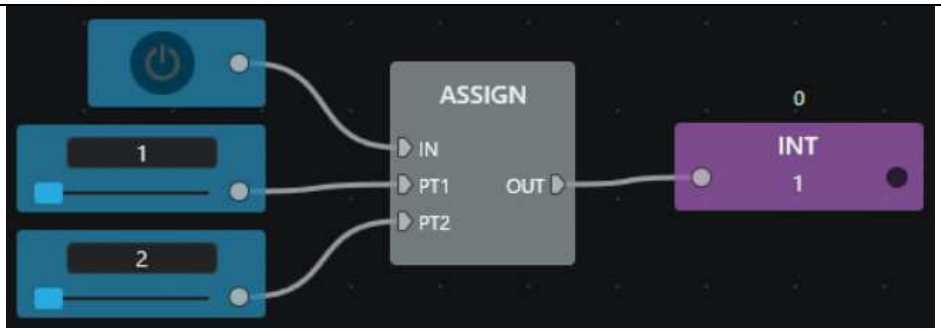
Входи	
CU	Запуск лічильника
RESET	Скидає значення CV до 0
PV	Початкове значення
Виходи	
Q	Q є TRUE, коли CV=PV. У всіх інших випадках Q є FALSE.
CV	Поточне значення

CTUD (Лічильник/ таймер)

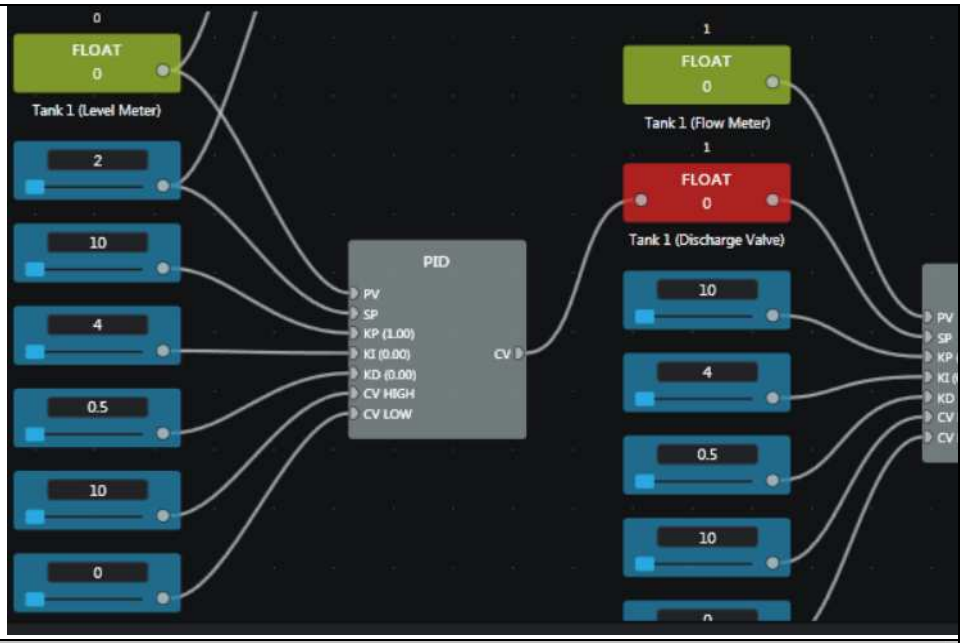
Входи	
CU	Запуск лічильника
CD	Запуск таймера
RESET	Скидає значення CV до 0
LOAD	Скидання таймера
PV	Початкове значення
Виходи	
QU	QU є TRUE, коли CV=PV. У всіх інших випадках QU є FALSE.
QD	QD є TRUE, коли CV=0. У всіх інших випадках Q є FALSE.
CV	Поточне значення

3.2.4.3 Extra (Особливі блоки) (табл. 3.19)

Таблиця 3.19 – Особливі функціональні блоки

<table>
<tr><th>Опис</th><th colspan="2">Візуальне зображення «сокета»</th></tr>
<tr><td rowspan="5">ASSIGN
OUT=PT2, якщо IN є TRUE, у всіх інших випадках OUT=PT1.</td><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Входи</td></tr>
<tr><td>IN</td><td>Умовне введення</td></tr>
<tr><td>PT1</td><td>Значення №-1</td></tr>
<tr><td>PT2</td><td>Значення №-2</td></tr>
<tr><td rowspan="13">DATETIME TO NUMERICAL
Перетворює DateTime в цифрові значення.</td><td colspan="2">Входи</td></tr>
<tr><td>IN</td><td>DateTime для перетворення</td></tr>
<tr><td colspan="2">Виходи</td></tr>
<tr><td>YEAR</td><td>Рік</td></tr>
<tr><td>MONTH</td><td>Місяць</td></tr>
<tr><td>DAY</td><td>День</td></tr>
<tr><td>HOURS</td><td>Години</td></tr>
<tr><td>MINUTES</td><td>Хвилини</td></tr>
<tr><td>SECONDS</td><td>Секунди</td></tr>
<tr><td>DAYOFYEAR</td><td>День року (від 0 до 366)</td></tr>
<tr><td>DAYOFWEEK</td><td>День тиждня (Sunday = 0, Saturday = 6)</td></tr>
<tr><td>TIMEOFDAY</td><td>Поточний час</td></tr>
<tr><td rowspan="11">NUMERICAL TO DATETIME
Перетворює числові значення в DateTime.</td><td colspan="2">Входи</td></tr>
<tr><td>YEAR</td><td>Рік</td></tr>
<tr><td>MONTH</td><td>Місяць</td></tr>
<tr><td>DAY</td><td>День</td></tr>
<tr><td>HOURS</td><td>Години</td></tr>
<tr><td>MINUTES</td><td>Хвилини</td></tr>
<tr><td>SECONDS</td><td>Секунди</td></tr>
<tr><td colspan="2">Виходи</td></tr>
<tr><td>OUT</td><td>Виводить значення DateTime</td></tr>
</table>

PID
Пропорційно – інтегруючо – диверенціюючий контролер.



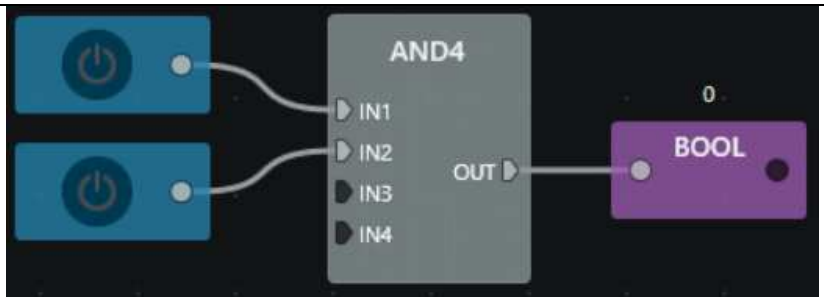
Входи	
PV	Вимірюване значення процесу
SP	Задана точка
KP	Пропорційний коефіцієнт (за замовчуванням = 1)
KI	Інтегральне підсилення
KD	Похідне підсилення
CV HIGH	Максимальне значення виводу
CV LOW	Мінімальне значення виводу
Виходи	
CV	Поточна змінна

3.2.4.4 Logical (Логічні блоки) (табл. 3.20)

Таблиця 3.20 – Логічні функціональні блоки

Опис	Візуальне зображення «сокета»
AND2 Виконує логічне І між двома булевими значеннями.	

AND4
Виконує логічне І між чотирма булевими значеннями.



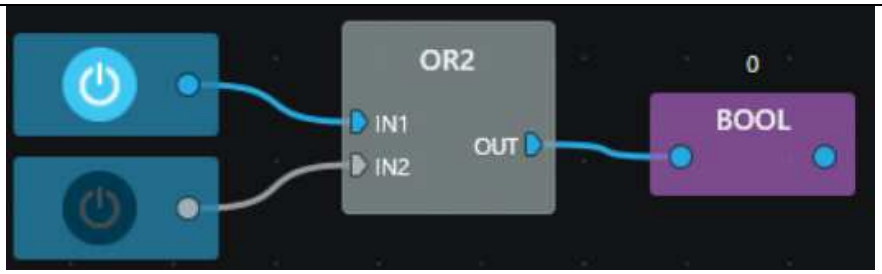
FTRIG
Виявляє перехід булевого значення від True до False.

JK
(JK триггер)
Коли входи J і K різні, вихід OUT приймає значення J.
Коли обидва входи є TRUE, то вихід вимикається.
Коли обидва входи є FALSE, вихід залишається увімкненим.

Входи	
J (SET)	Встановлює умову
CLK	Годинник
K (RESET)	Умови скидання
PRESET	Встановлює вихід
CLEAR	Очищує вихід
Виходи	
OUT	Значення пам’яті

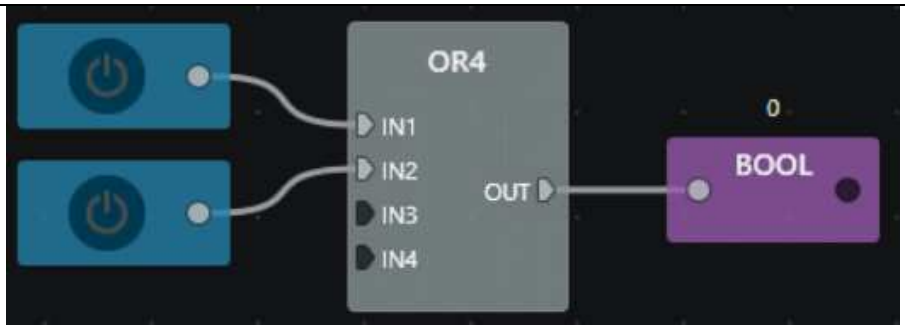
NOT
Доповнює булеве значення префіксом НЕ.

OR2
Виконує логічне АБО між двома булевими значеннями.



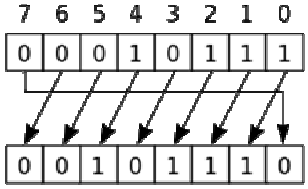
OR4

Виконує логічне АБО між чотирма булевими значеннями.



ROL

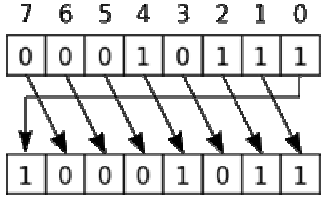
Виконує побітове ліве обертання на вказану кількість бітів. Приклад:



Входи	
IN	Число в якому будуть обертатися біти
ROT	Кількість бітів для оберту
Виходи	
OUT	Вихідне число після побітного лівого обертання

ROR

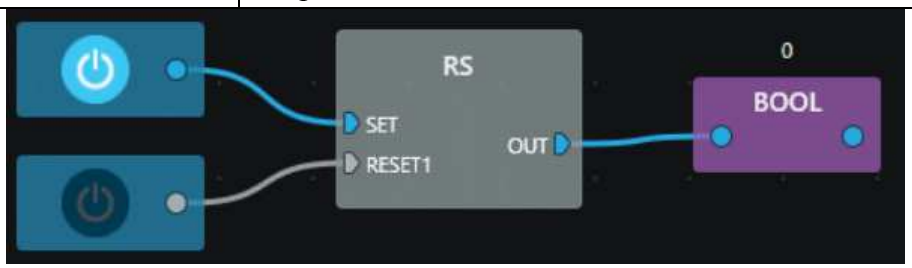
Виконує побітове праве обертання на вказану кількість бітів. Приклад:



Входи	
IN	Число в якому будуть обертатися біти
ROT	Кількість бітів для оберту
Виходи	
OUT	Вихідне число після побітного правого обертання

RS

Пам'ять із пріоритетом скидання.

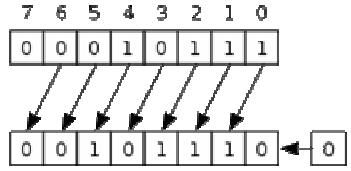


RTRIG

Виявляє перехід булевого значення від False до True.

SHL

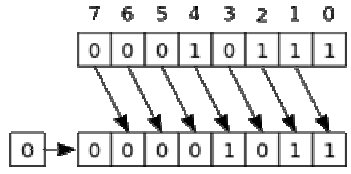
Виконує побітовий зсув ліворуч на вказану кількість бітів. Приклад:



Входи	
IN	Число в якому будуть зсуватись біти
SHIFTS	Кількість бітів для зсуву
Виходи	
OUT	Вихідне число після зсуву ліворуч

SHR

Виконує побітовий зсув праворуч на вказану кількість бітів. Приклад:



Входи	
IN	Число в якому будуть зсуватись біти
SHIFTS	Кількість бітів для зсуву
Виходи	
OUT	Вихідне число після зсуву праворуч

SR

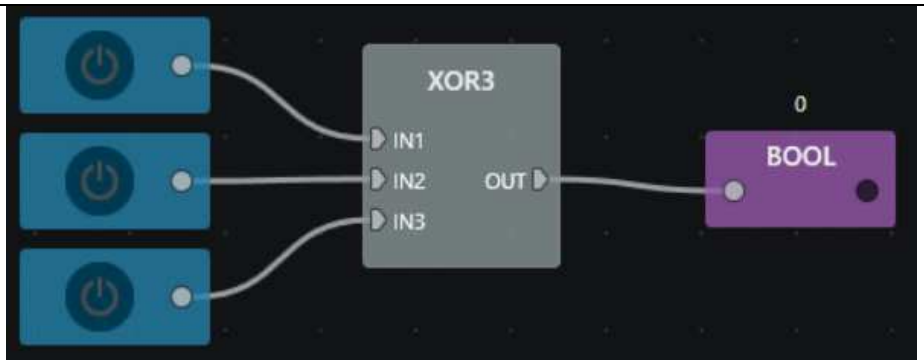
Пам'ять із заданим пріоритетом.

XOR2

Виконує логічне АБО між двома булевими значеннями.

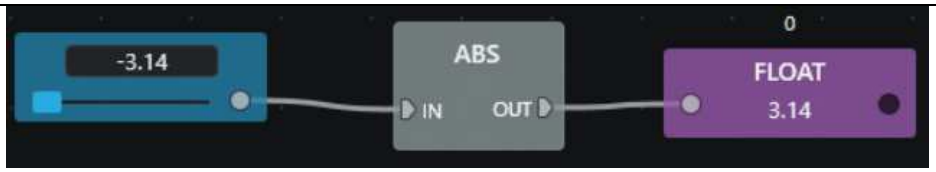
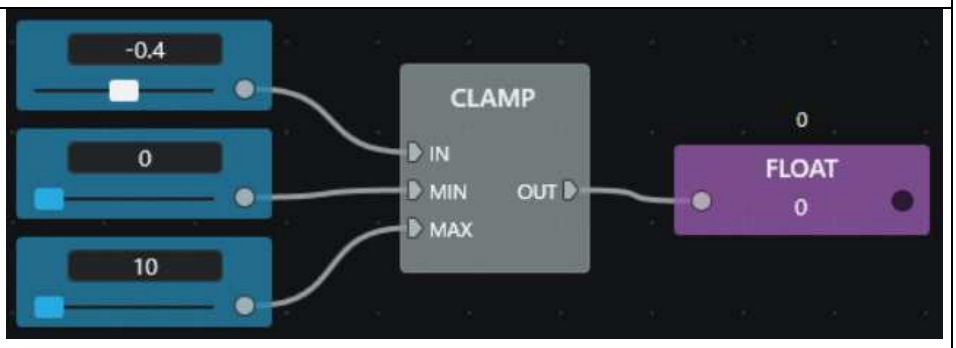
XOR3

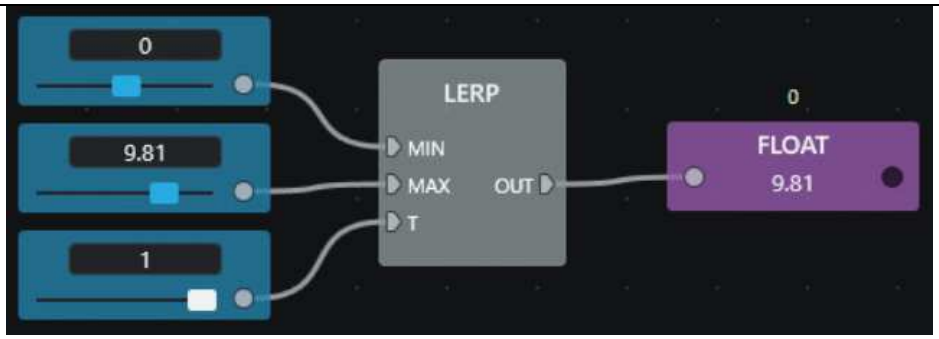
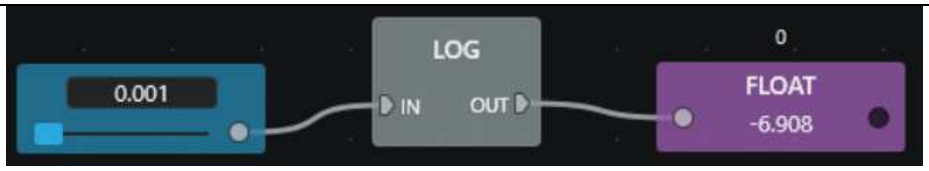
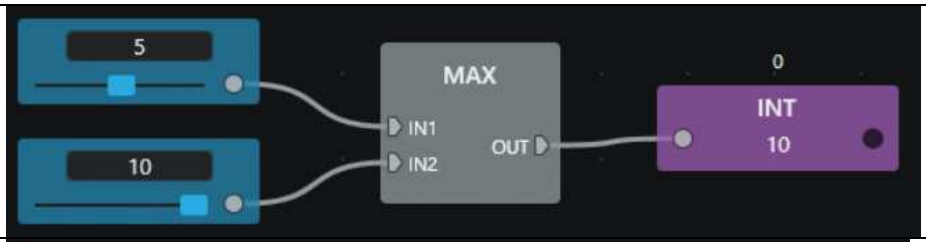
Виконує логічне АБО між трьома булевими значеннями.

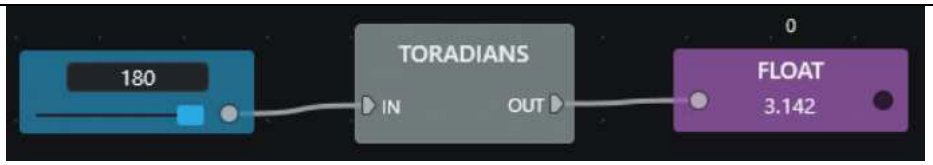


3.2.4.5 Math (Математика) (табл. 3.21)

Таблиця 3.21 – Математичні функціональні блоки

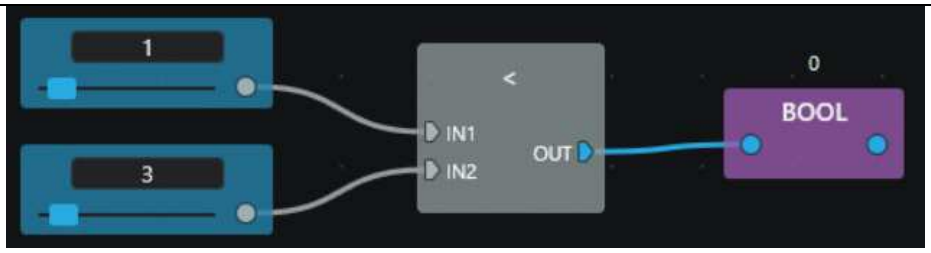
Опис	**Візуальне зображення «сокета»**
ABS Обчислює абсолютне значення вхідного числа	
ACOS Обчислює кут, косинусом якого є вхідне число.	
ASIN Обчислює кут, синусом якого є вхідне число.	
ATAN Обчислює кут, тангенсом до якого є вхідне число.	
CEILING Обчислює найменше ціле число, яке більше або рівне вхідному числу.	
CLAMP Задає мінімальне та максимальне допустиме значення для вихідного числа.	
COS Обчислює косинус вхідного кута (в радіанах).	
EXP Обчислює експоненту	

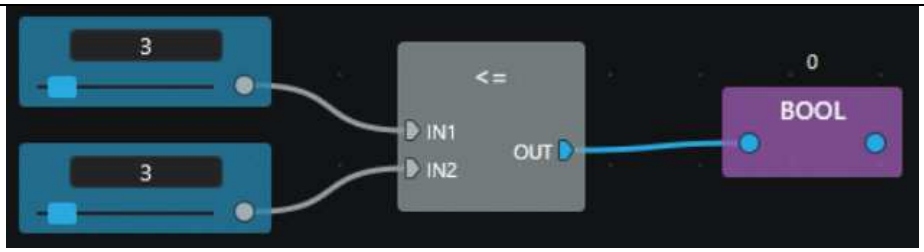
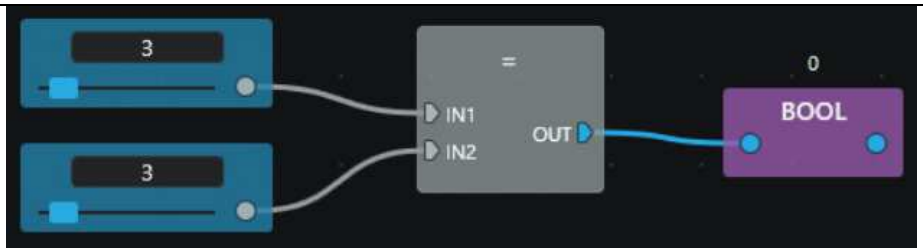
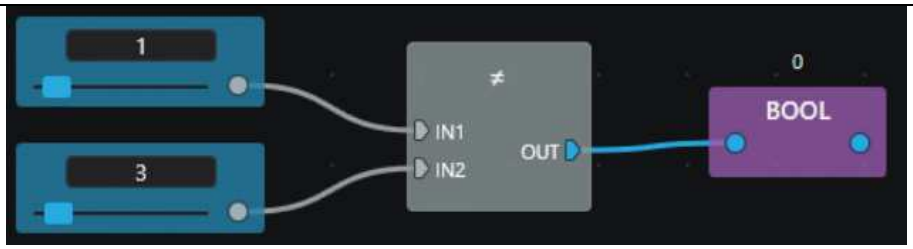
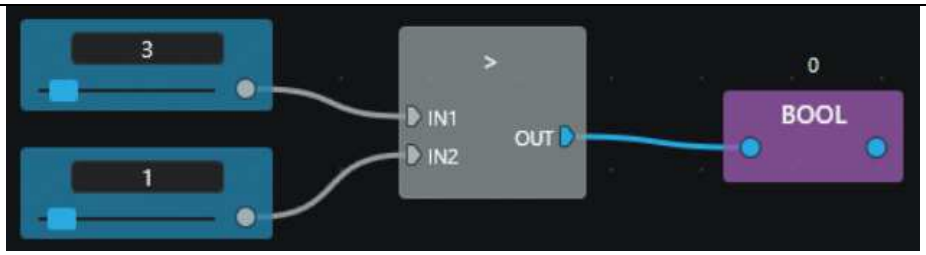
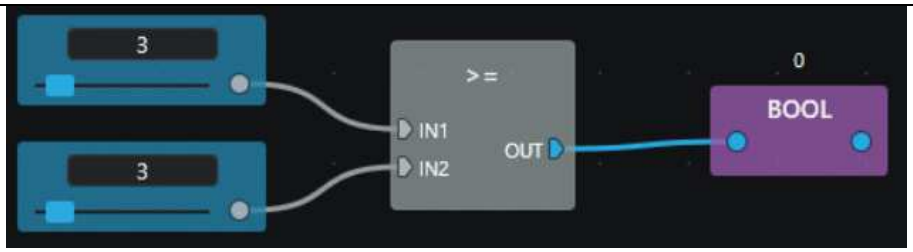
FLOOR Обчислює найбільше ціле число, менше або рівне вхідному числу.	
LERP Виконує лінійну інтерполяцію між двома значеннями. Вхідне значення T коливається від 0 до 1	
LOG Обчислює базовий логарифм вхідного значення.	
LOG10 Обчислює десятковий логарифм вхідного значення.	
MAX Визначає найбільше число з двох вхідних.	
MIN Визначає найменше число з двох вхідних.	
ROUND Заокруглює вхідне число до найближчого цілого значення.	

SIGN Визначає знак вхідного числа.	
SIN Обчислює синус вхідного кута (в радіанах).	
SQRT Обчислює квадратний корінь вхідного числа.	
TAN Обчислює тангенс вхідного кута (в радіанах).	
TODEGREES Переводить вхідне значення з радіанів у градуси.	
TORADIANS Переводить вхідне значення з градусів у радіани.	

3.2.4.6 Relational and Equality (Числові рівності і нерівності) (табл. 3.22)

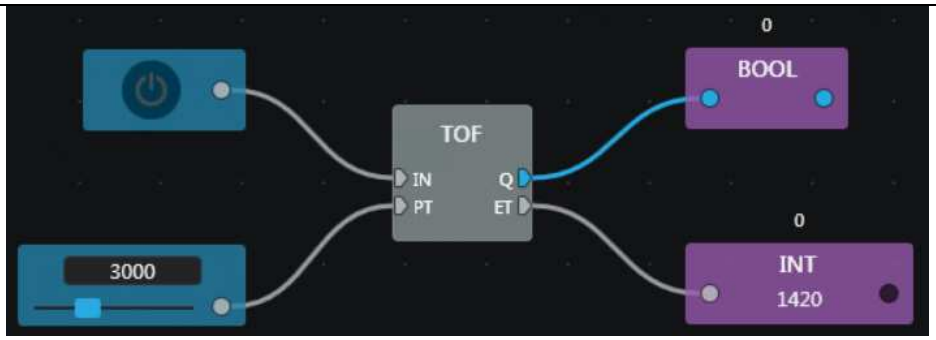
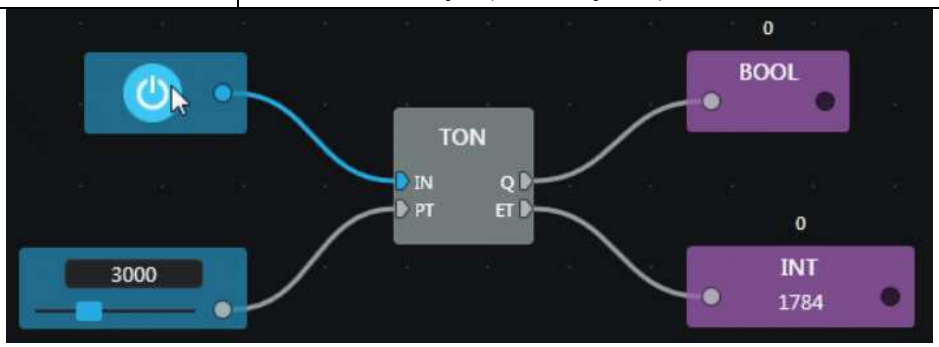
Таблиця 3.22– Числові рівності і нерівності

Опис	**Візуальне зображення «сокета»**
(<) **Менше** Визначає, чи перший операнд менший від другого.	

(<=) **Менше або рівне** Визначає, чи перший операнд менший або рівний другому.	
(=) **Дорівнює** Визначає, чи перший операнд дорівнює другому.	
(≠) **Не дорівнює** Визначає, чи перший операнд відрізняється від другого.	
(>) **Більше** Визначає, чи перший операнд більший, ніж другий.	
(>=) **Більше або рівне** Визначає, чи перший операнд більший або дорівнює другому.	

3.2.4.7 Timers (Таймери) (табл. 3.23)

Таблиця 3.23 – Функціональні блоки – таймери

<table>
<tr><td rowspan="5">TOF (таймер вимкнено)
Якщо IN=TRUE, то вихід Q=TRUE и вихід ET=0. Як тільки IN=FALSE, починається відлік часу на виході ET. При досягненні заданого значення, відлік зупиняється.</td><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Входи</td></tr>
<tr><td>IN</td><td>Запуск / скидання таймера</td></tr>
<tr><td>PT</td><td>Час який задається у мілісекундах</td></tr>
<tr><td colspan="2">Виходи</td></tr>
<tr><td rowspan="5">TON (таймер увімкнено)
Якщо IN= FALSE, то вихід Q= FALSE и вихід ET=0. Як тільки IN= TRUE, починається відлік часу на виході ET до значення рівного PT. При досягненні заданого значення, відлік зупиняється.</td><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Входи</td></tr>
<tr><td>IN</td><td>Запуск / скидання таймера</td></tr>
<tr><td>PT</td><td>Час який задається у мілісекундах</td></tr>
<tr><td colspan="2">Виходи</td></tr>
<tr><td></td><td>Q</td><td>Вихідне значення</td></tr>
<tr><td></td><td>ET</td><td>Час, який минув (мілісекунди)</td></tr>
</table>

3.3 Висновок

У цьому розділі було розглянуто та систематизовано функціональні можливості таких середовищ, як FACTORY I/O та CONTROL I/O. Було розглянуто та наочно показано всі компоненти та елементи, за допомогою яких відбувається моделювання обладнання та які доступні у цих двох програмних засобах. Також було продемонстровано основні методи та способи взаємодії з даними програмними продуктами.

4 КОНСТРУКТОРСЬКА ЧАСТИНА

4.1 Загальні поняття про автоматне програмування та доцільність його використання у лабораторному комплексі

4.1.1 Парадигма програмування

Парадигма програмування — це система ідей і понять, які визначають стиль написання комп'ютерних програм, а також спосіб мислення програміста. Також, це — спосіб концептуалізації, що визначає організацію обчислень і структурування роботи, яку виконує комп'ютер.

Парадигма програмування унаочнює те, як програміст розглядає роботу програми; наприклад, за ООП (Об'єктно-орієнтоване програмування) — як множини об'єктів, тоді як за ФП (Функційне програмування) — як послідовності обчислень функцій без станів.

Кожну окрему парадигму програмування характеризує наявність у ній методу та зв'язку із моделлю життєвого циклу. Спільним для різних парадигм програмування є загальні принципи проектування програмного продукту. Користувач може вибирати ту або іншу парадигму програмування з позицій зручності застосування для виготовлення конкретного програмного продукту.

Важливо відзначити, що парадигма програмування не визначається однозначно мовою програмування; практично всі сучасні мови програмування в тій чи іншій мірі допускають використання різних парадигм (мульти-парадигмальність програмування).

4.1.2 Дискретний автомат

Цифровий (дискретний) автомат – пристрій, який здійснює прийом, зберігання та перетворення дискретної інформації за деяким алгоритмом. В якомусь розумінні до автоматів можна віднести як реальні пристрої (обчислювальні машини, живі організми і т.д.), так і абстрактні системи (математичні машини, аксіоматичні теорії та ін.).

Загальну теорію автоматів підрозділяють на абстрактну і структурну. Відмінність між ними полягає в тому, що абстрактна теорія, відсторонюючись від

структури автомата (тобто не цікавлячись способом його побудови), вивчає лише поведінку автомата відносно зовнішнього середовища. Абстрактна теорія автоматів близька, таким чином, до теорії алгоритмів, будучи, по суті, її подальшою деталізацією. В протилежність абстрактній теорії структурна теорія цікавиться як структурою самого автомата, так і структурою вхідних дій і реакцією автомата на них. У структурній теорії вивчаються способи побудови автоматів, способи копіювання вхідних дій і реакцій автомата. Таким чином, структурна теорія автоматів являється продовженням і подальшим розвитком абстрактної теорії. Спираючись на апарат булевих функцій і на абстрактну теорію автоматів, структурна теорія дає ефективні рекомендації щодо розробки реальних пристроїв обчислювальної техніки. Найвідомішим прикладом де використовуються автомати є машина Тюрінга.

Автомат складається з: вхідної стрічки, керуючого пристрою, допоміжної або робочої пам'яті. Приклад взаємодії цих елементів можна побачити на наступному рисунку (рис. 4.1).

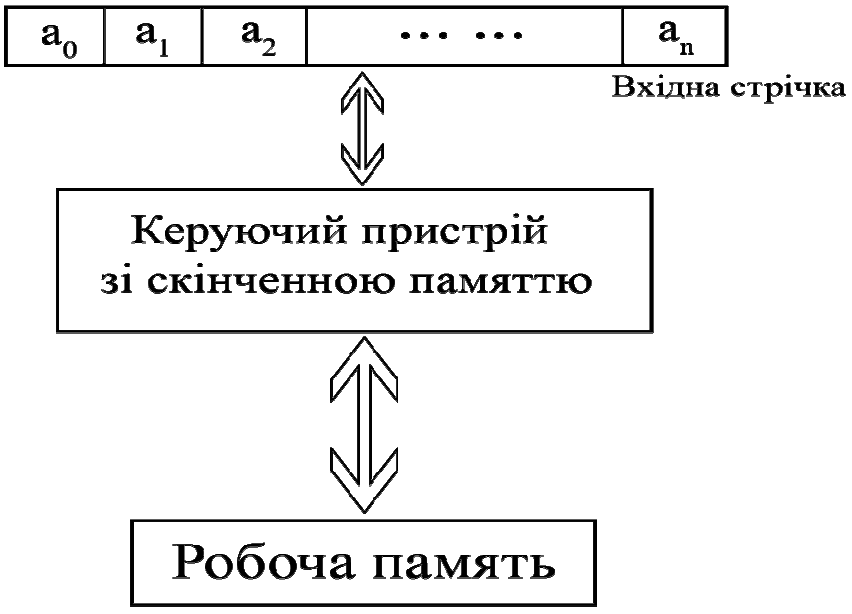


Рисунок 4.1 Взаємодія елементів автомата

Вхідну стрічку можна розглядати як лінійну послідовність кліток, або комірок, причому кожна комірка точно містить один вхідний символ з деякого

скінченного вхідного алфавіту. За допомогою вхідної стрічки зображується інформація, що поступає на вхід автомата.

Вхідна голівка у кожен момент читає одну вхідну комірку. За один крок роботи автомата вхідна голівка може зміститися на одну комірку вліво, вправо або залишитися нерухомою. Автомат, що не переміщує свою голівку називається одностороннім.

Пам'ять автомата – структура, в якій записуються, зберігаються і зчитуються додаткові дані, що використовуються автоматом при роботі. Для кожного виду автоматів строго визначено тип пам'яті.

Робота автомата складається з послідовності тактів. Кожен такт складається з таких дій:

1. Читається поточний вхідний символ.
2. За допомогою функції доступу досліджується пам'ять, і до неї заноситься деяка інформація.
3. Змінюється стан керуючого пристрою.
4. Записується вихідна інформація у комірки вхідної стрічки.
5. Вхідна голівка зміщується на одну комірку вліво, вправо або зберігається у початковому стані.

Поточний вхідний символ та інформація, що витягнута з пам'яті, разом з поточним станом керуючого пристрою визначають, яким повинен бути цей такт. Таким чином, на кожному такті визначається поточна конфігурація автомата, до якої входять:

- поточний стан керуючого пристрою;
- поточний зміст вхідної стрічки;
- поточний зміст робочої пам'яті.

Конфігурація автомата змінюється на кожному такті під впливом керуючого пристрою.

Керуючий пристрій складається зі скінченної множини станів $S = \{s_0...s_n\}$ і відображень, які залежно від попередньої конфігурації дозволяють визначити

нову конфігурацію автомата, напрямок переміщення вхідної голівки(якщо вона не одностороння), інформацію на друку(якщо в автоматі передбачено функцію друку), інформацією, що заносять у пам'ять і витягують звідти(якщо автомат має робочу пам'ять). Є два види керуючого пристрою:

- Недетермінований – для кожної конфігурації існує скінченна множина можливих конфігурацій(більше однієї), так що в будь-яку з них автомат може перейти на наступному кроці.
- Детермінований – для кожної конфігурації існує не більше однієї можливої наступної конфігурації.

4.1.3 Автоматне програмування

Автоматне програмування — парадигма програмування, відповідно до якої програма або її фрагмент розглядаються як модель деякого формального автомата (покрокового перетворювача інформації), а вибір операції обробки вхідних даних визначається не тільки поточним значенням даних, але й поточним значенням стану автомата.

Залежно від конкретної задачі в автоматному програмуванні можуть використовуватися як кінцеві автомати, так і автомати більш складної структури. Визначальними для автоматного програмування є наступні особливості:

1. Послідовність операцій виконання програми розбивається на кроки автомата, кожен з яких являє собою виконання певної (однієї і тієї ж для кожного кроку) секції коду з єдиною точкою входу; така секція може бути оформлена, наприклад, у вигляді окремої функції і, в свою чергу, може бути розділена на підсекції, відповідно до окремих станів або категорій станів;
2. Збереження інформації про стан між кроками автомата здійснюється виключно через явно визначений набір змінних, призначених для зберігання стану автомата, таких, що стан програми на будь-якому окремому моменті входу в крок автомата можуть різнитися між собою виключно значеннями цих і тільки цих змінних.

3. Зміна станів автомата відбувається в циклі (можливо, неявно заданому) виконання кроків автомата.

Основним поняттям в автоматному програмуванні є стан. При написанні автоматних програм пропонується розділяти стани на два класи: керуючі і обчислювальні. При цьому за допомогою невеликого числа керуючих станів, як і в машині Тюрінга, можна управляти як завгодно великим числом обчислювальних станів. У введеній класифікації керуючі стани можуть бути названі якісними, а обчислювальні – кількісними. В рамках автоматного програмування основна увага приділяється керуючим станам, які і будуть розглядаються в подальшому.

При цьому справедливе співвідношення: «Стан + вхідний вплив = кінцевий автомат без виходу». Також справедливо: «Автомат без виходу + вихідні впливи = автомат».

Автомати можуть бути абстрактними (вхідні та вихідні впливи формуються послідовно) і структурними (вхідні та вихідні впливи формуються «паралельно»). У автоматному програмуванні, на відміну, наприклад, від програмування компіляторів, зазвичай застосовуються структурні автомати. Час в автоматах в явному вигляді не використовується. При потребі застосовуються елементи затримки, вони розглядаються як об'єкти управління. При цьому затримки запускаються і скидаються з автоматів, а інформація про закінчення часу надходить до них у вигляді вхідних впливів.

Автомати можуть задаватися в різному вигляді, проте при їх проектуванні та використанні людиною, вони повинні володіти когнітивними властивостями, що досягається при заданні поведінки автоматів у вигляді графів переходів (діаграм станів). У разі якщо автомати генеруються автоматично, то вони можуть задаватися інакше, наприклад, в табличній формі.

При цьому потрібно відзначити, що навіть при порівняно невеликому числі станів і переходів, таке завдання ускладнює розуміння роботи автоматів людиною, так як відображення переходів в них не є наочним.

Зрозумілість графів переходів досягається за рахунок того, що стани всіх вхідних впливів відповідного автомата декомпозуются на групи, кожна з яких визначає переходи з розглянутого стану.

4.1.4 Переваги автоматного програмування

До сильних сторін АП можна віднести наступне:

1. Простота і, як наслідок, надійність. Йдеться про те, що прагнення триматися рамок єдиного автоматного формалізму змушує нас більш строго ставитися до створюваного програмного коду. Цей код будується з невеликих за обсягом «будівельних» елементів. В цьому і полягає простота. Короткі, уніфіковані, побудовані за єдиним принципом процедури легше контролювати.
2. Наочність алгоритму. Складність алгоритму поведінки нікуди не ділась. Просто вона перенесена з програмного коду ззовні, в той малюнок – граф переходів (діаграму станів), - яким користується програміст, реалізуючи автомат. А з програмної точки зору сам автомат разом з логікою його роботи представлений простою структурою – матрицею переходів. Аналізувати поведінку робота, виходячи з графа автомата дійсно зручно і наочно. Це в якійсь мірі – особливості психології людського сприйняття.
3. Гнучкість алгоритму. Автоматне уявлення дозволяє поступово нарощувати складність і гнучкість поведінки робота, не ламаючи при цьому логіку роботи всієї програми, а це – вкрай важливо.

4.2 Різновиди автоматного програмування

Автоматне програмування розвивається в трьох основних напрямках: логічне управління, програмування з явним виділенням станів і об'єктно-орієнтоване програмування з явним виділенням станів.

4.2.1 Логічне управління

Найбільш природньо застосовувати автомати в системах логічного управління, в яких вхідні і вихідні змінні є двійковими (0 та 1). Природність застосування автоматів при програмуванні цього класу систем пояснюється тим, що програми в них заміняються логічними схемами, проектування яких з використанням автоматів є поширеним і розвивається давно, починаючи з релейно-контактних схем.

Однак простота і природність застосування автоматів, мабуть, далеко не очевидна, так як навіть при програмуванні логічних контролерів, практично ніхто не пропонував спочатку проектувати автомати, а потім реалізовувати їх на обраній мові програмування.

4.2.2 Програмування з явним виділенням станів

Більш широким класом в порівнянні з системами логічного управління є реактивні системи. Для опису поведінки таких систем, куди входить, більшість вбудованих систем, застосування автоматів також природньо, як і для систем логічного управління. Реактивні системи є більш складними в порівнянні з системами логічного управління. У них:

- в якості вхідних впливів, поряд з вхідними змінними, використовуються стани;
- запуск програм здійснюється по станах, а не циклічно;
- в якості вихідних впливів можуть використовуватися не тільки виконавчі, але й інші функції, що дозволяє називати автомати, що застосовуються – гібридними;

- автомати можуть містити не тільки вкладені стани, а й вкладені автомати;
- автомати можуть взаємодіяти не тільки за рахунок перевірки номерів станів, як було запропоновано для систем логічного управління, але також і за рахунок вкладеності, яка викликана обміном подіями (повідомленнями).

4.2.3 Об'єктно-орієнтоване програмування з явним виділенням станів

Вже два десятиліття об'єктно-орієнтоване програмування (ООП) є найбільш широко використовуваним стилем програмування в світі. При застосуванні об'єктної парадигми програми будують з об'єктів, що взаємодіють за рахунок обміну повідомленнями.

В теорії об'єктно-орієнтованого програмування вважається, що об'єкт має внутрішній стан і здатний отримувати повідомлення, відповідати на них, відправляти повідомлення іншим об'єктам і в процесі обробки повідомлень змінювати свій внутрішній стан. Більш наближене до практики, поняття виклику методу об'єкта, вважається синонімом поняття відправки повідомлення об'єкту.

Таким чином, з одного боку, об'єкти в об'єктно-орієнтованому програмуванні можуть розглядатися як кінцеві автомати (або, якщо завгодно, моделі кінцевих автоматів), стан яких являє собою сукупність внутрішніх полів, в якості кроку автомата можуть розглядатися один або кілька методів об'єкта при умови, що ці методи не викликають ні самі себе, ні один одного ні прямо, ні побічно.

З іншого боку, очевидно, що поняття об'єкта є вдалим інструментом для реалізації моделі кінцевого автомата. При застосуванні парадигми автоматного програмування в об'єктно-орієнтованих мовах зазвичай моделі автоматів представляються у вигляді класів, стан автомата описується внутрішніми (закритими) полями класу, а код кроку автомата оформляється у вигляді методу класу, причому такий метод швидше за все виявляється єдиним відкритим методом (не рахуючи конструкторів і деструкторів), що змінюють стан автомата. Інші відкриті методи можуть служити для отримання інформації про стан

автомата, але не змінюють його. Всі допоміжні методи (наприклад, методи-обробники окремих станів чи їх категорій) в таких випадках зазвичай прибирають в закриту частину класу.

4.3 Реалізація автоматного програмування

Процес реалізації автоматного програмування можна розбити на 3 етапи:

1) Виділення з системи об’єктів, які можуть відповідати фізичним об’єктам, їх групам або їх частинам. В процесі такого виділення необхідно дотримуватись мети досягнення найбільш простого вигляду системи та найбільш простого її подальшого програмування. Зазвичай можна притримуватись правила: «один елемент системи - один об’єкт – одна змінна стану».
2) Після декомпозиції системи необхідно виділити стани кожного об’єкта. Бажано при цьому щоб зміна вихідних сигналів для управління об’єктами здійснювалась лише при змінної стану об’єкта.
3) Після виділення станів всіх об’єктів необхідно виділити умови переходу об’єкта з одного стану в другий. Такі умови можуть визначатись станом датчиків або набуттям якимось іншим об’єктом заданого свого стану.

4.3.1 Приклад задачі із одною змінною стану

Розглянемо просту задачу із використанням одного об’єкта та одної змінної стану. Нехай на конвеєрі потрібно згрупувати 3 коробки (рис. 4.2), які будуть відправлені на вихід. Коробки поступають незалежно з вхідної позиції і розміщуються на вхідному конвеєрі. По мірі надходження коробки передаються на буферний конвеєр, де накопичуються у кількості 3 штуки. Після накопичення (3 коробки) вони передаються на вихід.

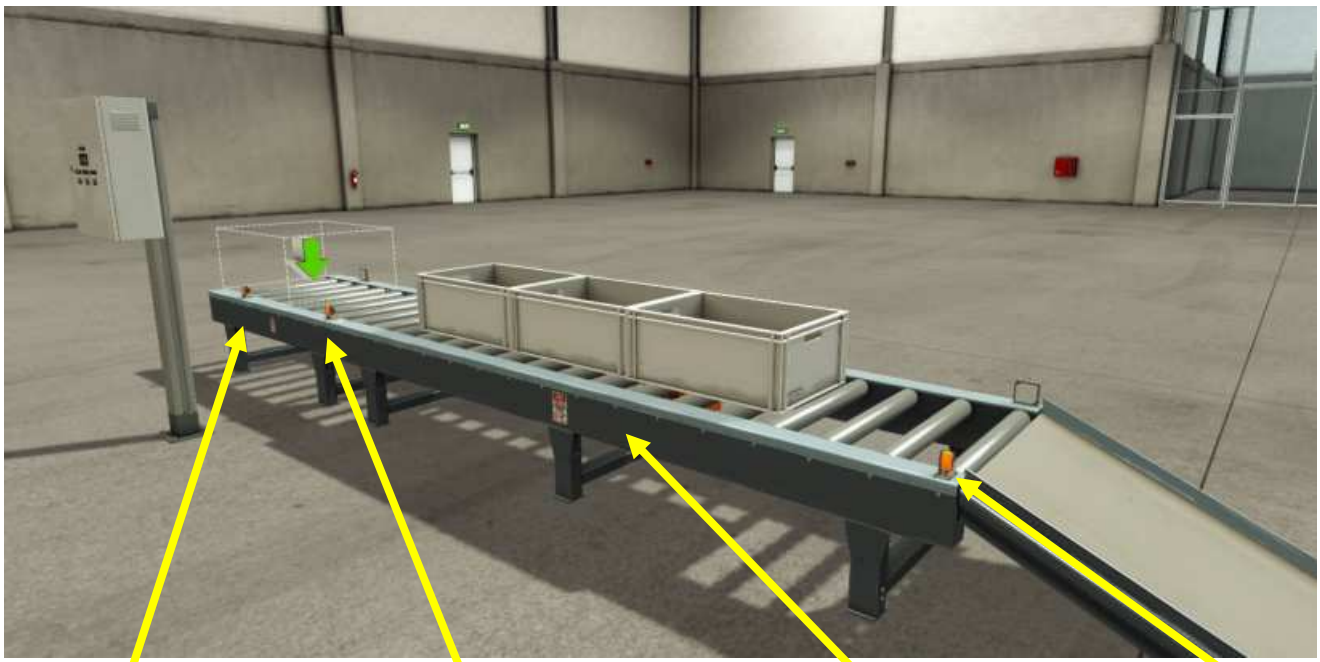


Рисунок 4.2 Приклад сцени у FACTORY I/O

В даній задачі будемо використовувати один об’єкт та одну змінну стану котра його описує. Легко зрозуміти, що в системі можливі 5 станів:

0) Всі конвеєри виключені. Цей стан є початковим. Вихід з даного стану здійснюється при запуску програми.

1) Жодної коробки на буферному конвеєрі немає, система очікує надходження першої коробки. В даному стані вхідний конвеєр має бути включений, буферний виключений. Умовою виходу з даного стану буде спрацювання датчика (D1) на виході вхідного конвеєра, що буде вказувати на появу коробки на буферному конвеєрі. В такому випадку система переходить у стан 2.

2) У стані 2 коробка передається з вхідного конвеєра на буферний, виходом із даного стану буде відключення датчика (D1) на виході вхідного конвеєра. З стану 2 можливі переходи в стан 1 та в стан 3. Якщо на буферному конвеєрі находиться менше 3-х коробок, то здійснюється перехід на завантаження

наступної коробки. Якщо ж на буферному конвеєрі є 3 коробки , система переходить у стан 3.

3) У 3 стані вхідний конвеєр виключається і здійснюється розвантаження буферного конвеєра. З 3 стану здійснюється перехід у 1 стан після вивантаження останньої коробки, що контролюється переключенням датчика (D2) встановленого на виході буферного конвеєра. В стан 4 здійснюється перехід у тому випадку коли датчик (D2) спрацьовує, але на конвеєрі ще залишились коробки.

4) В стані 4 продовжується вивантаження коробки з буферного конвеєра і здійснюється перехід у стан 3 після проходження коробки біля датчика.

Тобто система спочатку 3 рази переходить між станами 1 та 2, накопичуючи коробки на буферному конвеєрі, а потім шляхом переходів між станами 3 і 4 здійснює їх вивантаження. Фактично система працює між станами 1-2-1-2-1-2-3-4-3-4-3-4, а потім знову повторює цикл. При реалізації такого циклу нам необхідно буде мати лічильник кількості коробок, котрий буде збільшуватись на 1 (k+1) при переході між станами 1 та 2. І зменшуватись на 1 (k-1) при переході між станами 3 і 4. Діаграму станів до нашого прикладу можна переглянути нижче (рис. 4.3).

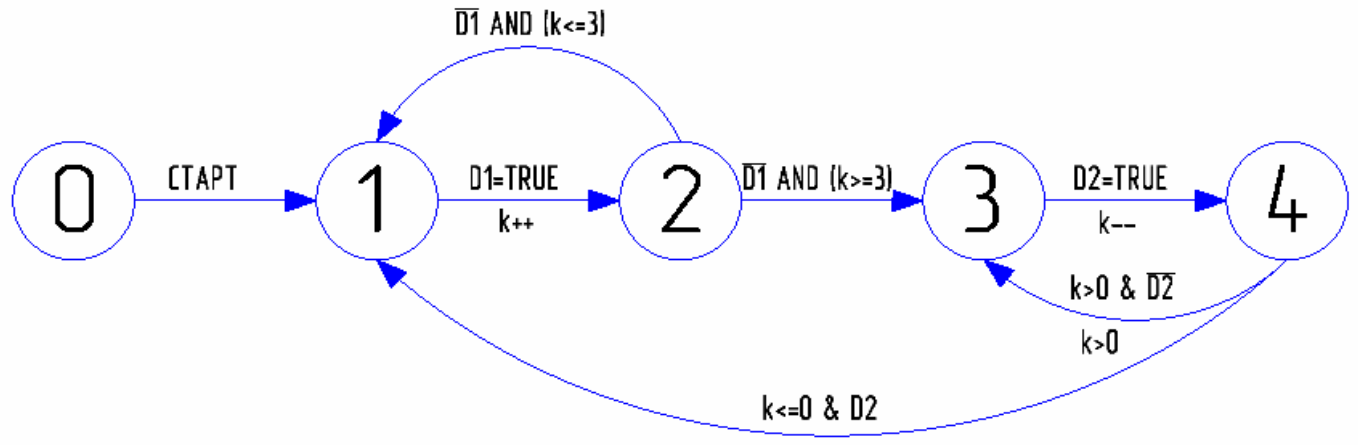


Рисунок 4.3 Діаграма станів до прикладу задачі із одною змінною стану

Як видно з нашого опису крім змінних стану при автоматному програмуванні можна використовувати довільні змінні, котрі тим чи іншим чином визначають параметри об’єктів. В даному випадку такою додатковою змінною є лічильник кількості коробок.

4.3.2 Класичний підхід до програмування

В даному випадку роботу системи можна також описати у вигляді класичного алгоритму (додаток А).

Як видно з рисунку алгоритм є досить складним заплутаним і тяжким для модифікації. Тому опис роботи системи у вигляді алгоритму в даному випадку є незручним.

4.3.3 Програмування з використання змінної стану

Найпростіший приклад реалізації на мові ST вказаної логіки роботи нашої системи показаний у додатку Б. Як видно основна управляюча змінна котра визначає стан системи реалізується в операторі case. Кожен підблок оператора case розділяється на 2 частини. У першій частині вказується, що необхідно включити і виключити для отримання даного стану. Наприклад включити вхідний конвеєр, виключити вихідний конвеєр. У другій частині вказується чи визначається умова виходу із даного стану, стан в який необхідно перейти, а також дії котрі здійснюються при даному переході. Наприклад збільшення або зменшення лічильника на 1 тощо. Умови можуть бути вкладеними якщо це потрібно (приклад умови реалізується у стані 2).

Лічильник збільшується та зменшується лише при переходів між станами, тому оператор збільшення та зменшення знаходиться у програмі в частині де здійснюється аналіз можливості переходу у новий стан.

4.4 Висновок

У цьому розділі було закладено теоретичні та методологічні основи для виконання задач поставлених у лабораторних роботах. Також було порівняно основні напрямки автоматного програмування, а саме: логічне управління, програмування з явним виділенням станів і об'єктно - орієнтоване програмування з явним виділенням станів. Під час аналізу існуючих методів програмування, було встановлено, що автоматне програмування має ряд переваг перед класичним підходом до програмування та програмуванням з використанням змінної стану.

5 СПЕЦІАЛЬНА ЧАСТИНА

5.1 Сценарій для проведення лабораторної роботи на тему: «Ознайомлення з основами роботи у середовищі програмного забезпечення «FACTORY I/O» та запуск готового проекту»

Дана лабораторна робота виконується у двох середовищах (програмах): CODESYS та FACTORY I/O.

Потрібно запустити CODESYS Control Win V3 та CODESYS V3.5 SP10. Далі необхідно відкрити готову програму пройшовши по шляху: (File / Open Project (ctrl+O) / DATA(D:) / FACTORY_I_O / Laboratorna_robota_№_21 / Codesys / Factory_Lab_21). Код програми можна переглянути у додатку В. Запуск готової програми можна здійснити натисканням на клавішу Login у вкладці Online.

Наступним кроком є запуск програми FACTORY I/O та відкриття готової сцени: File / Open / My Scenes / Laboratorna_robota_№_21.

Необхідно натиснути на піктограму  , як показано на рисунку 5.1.

Рисунок 5.1 Вибір драйвера

Далі з випадаючого вікна DRIVER слід вибирати OPC Client Data Access. Якщо все правильно зроблено то драйвер повинен мати вигляд, як на рисунку 5.2.

Рисунок 5.2 Готове вікно драйвера

Далі слід закрити вікно драйвера, натиснути на  і на  . Спостерігати за візуалізацією процесу роботи обладнання.

5.2 Сценарій для проведення лабораторної роботи на тему: «Модифікація та відлагодження проекту у середовищі програмного забезпечення «FACTORY I/O»-CODESYS»

У попередній лабораторній роботі було розглянуто готовий проект для програмного середовища «FACTORY I/O» та підключення його 3D-моделі до віртуального контролера. Керування конвеєром здійснювалось програмно, програма написана мовою ST (Structured text) у середовищі CODESYS.

У цій лабораторній роботі необхідно здійснити модифікацію базового проекту (який розглядався у підрозділі 5.1), код до якого подано у Додатку В. Слід налагодити сцену на коректну роботу за допомогою внесення змін у програмний код, написаний мовою ST. Варіанти завдань наведені у таблиці 5.1.

Таблиця 5.1 – Варіанти завдань

Варіант	**Завдання**
1	Сортування коробок на палетах за висотою (високі рухаються прямо, низькі повертають праворуч)
2	Почергове подаванн коробок на палетах по 2 штуки у гілки конвеєра
3	Подавання коробок на палетах почергово: 2 штуки прямо, 1 праворуч
4	Подавання коробок на палетах прямо у послідовності одна висока, одна низька, інші коробки направляються праворуч
5	Подавання коробок на палетах у послідовності 1 штука прямо, 2 штуки праворуч
6	Сортування коробок на палетах за висотою (низькі рухаються прямо, високі повертають праворуч)

5.3 Сценарій для проведення лабораторної роботи на тему: «Розробка та відлагодження програми керування технологічним обладнанням у середовищі програмного забезпечення «FACTORY I/O»-CODESYS»

У цій лабораторній роботі показано, як використовувати CODESYS разом із «FACTORY I/O» для моделювання роботи автоматизованаого обладнання на базі OPC Data Access. Слід створити новий проект автоматизованого обладнняння з відлагодженням у середовищі CODESYS програми керування для програмованого логічного контролера. Як приклад буде розглядатись сцена - «Sorting by Height».

Необхідно запустити ПЗ «FACTORY I/O» та відкрити готову сцену з технологічним обладнанням з бібліотеки. На панелі інструментів слід з меню File вибрати Open. У меню вибору сцен натиснути ЛКМ на Scenes. Обрати сцену «Sorting by Height» (рис. 5.3).

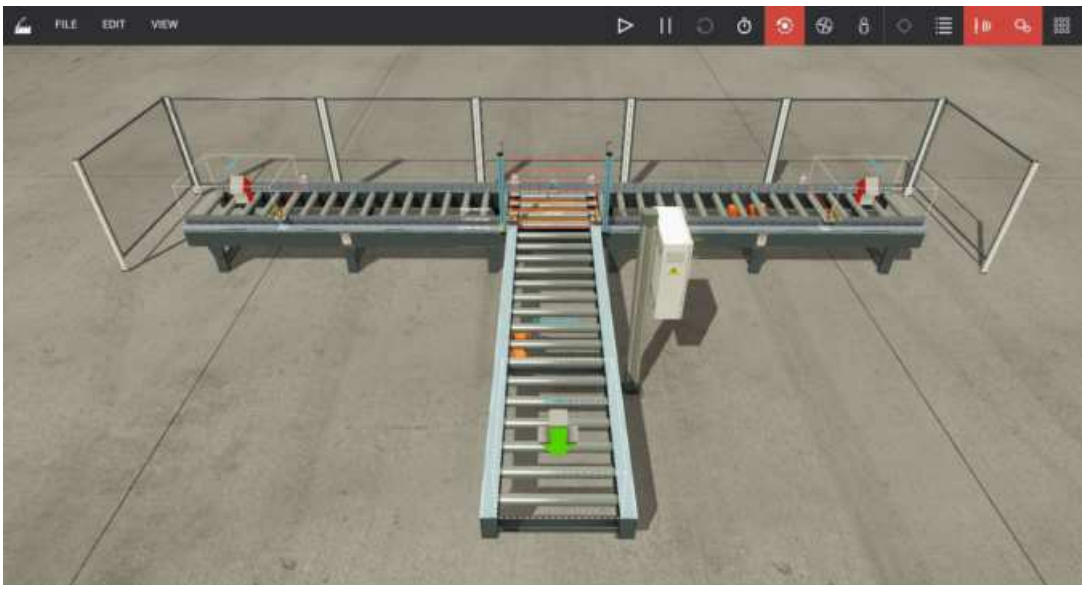


Рисунок 5.3 Сцена «Sorting by Height»

Вивчити принцип роботи технологічного обладнання, поданого у сцені, визначити місцезнаходження та типи приводів і датчиків, їх позначення та змінні, пов’язані з ними. Результати аналізу занести у таблиці (див. табл. 5.3 і 5.4).

Таблиця 5.3 – Приводи і виконавчі механізми

№ п/п	Призначення	Позначення на сцені	Позначення змінної у програмі	Тип змінної	Стан змінної при запуску
1	Центральний роликовий конвеєр (Переміщує палети)	Conveyor entry	oConveyorEntry	BOOL	True
2	Правий роликовий конвеєр (Переміщує палети)	Conveyor right	oConveyorRight	BOOL	True
3	Лівий роликовий конвеєр (Переміщує палети)	Conveyor left	oConveyorLeft	BOOL	True
4	Транспортер ланцюгового типу (Направляє палету на потрібний конвеєр)	Transf. right Transf. left	oTransfRight oTransfLeft	BOOL BOOL	False False

Таблиця 5.4 – Датчики

№ п/п	Призначення	Позначення на сцені	Позначення змінної у програмі	Тип змінної	Стан змінної при запуску
1	Давач для вимірювання висоти коробок	High sensor Low sensor	iHigh_sensor iLow_sensor	BOOL BOOL	False False
2	Давач виходу палети з ланцюгового транспортера	At right entry At left entry	iAtRightEntry iAtLeftEntry	BOOL BOOL	False False
3	Давач виходу палети з конвеєра	At right exit At left exit	iAtRightExit iAtLeftExit	BOOL BOOL	False False

Дати словесний опис принципу роботи автоматизованого обладнання.

Приклад:

При переміщенні коробок на палетах по центральному конвеєрі, здійснюється вимірювання висоти кожної коробки. Коли палета потрапляє на транспортер ланцюгового типу, він направляє її в потрібну сторону (направо або наліво) згідно із спрацюванням давачів висоти.

Конструктивно транспортер складається із двох приводів – ланцюгового та роликового, що дозволяє йому переміщати об’єкти у чотирьох напрямках. У стані по замовчуванні роликовий привід знаходиться вище ланцюгового (рис. 5.4), тобто об’єкти рухаються або вперед або назад. Коли є потреба у переміщенні об’єктів вправо або вліво , ланцюговий привід піднімається (рис. 5.5).

Коли палета потрапляє на один з конвеєрів, давачі (At right entry або At left entry) вказують транспортеру, що на ньому немає палети. Тоді ланцюговий привід опускається, для того щоб на нього могла потрапити нова палета.

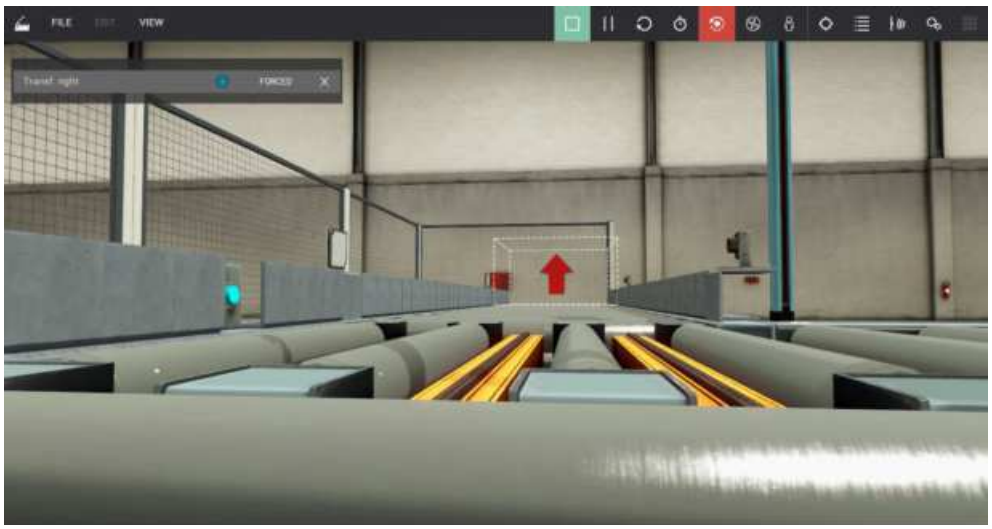


Рисунок 5.4 Ланцюговий транспортер у стані по замовчуванню

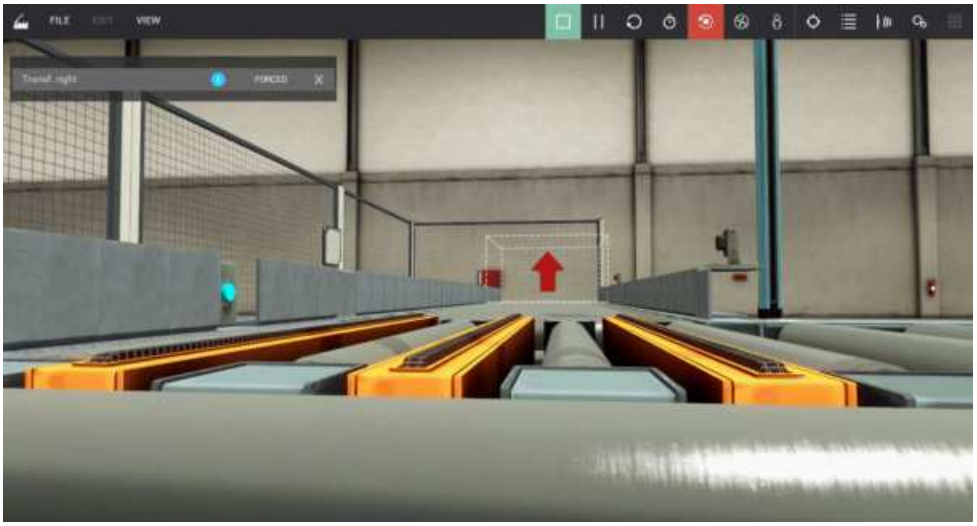


Рисунок 5.5 Ланцюговий транспортер з піднятим приводом

На основі словесного опису розробити блок-схему алгоритму роботи обладнання. Приклад блок-схеми алгоритму подано у додатку Г.

Після виконня всіх цих пунктів слід запустити CODESYS та створити новий проект (рис. 5.6).

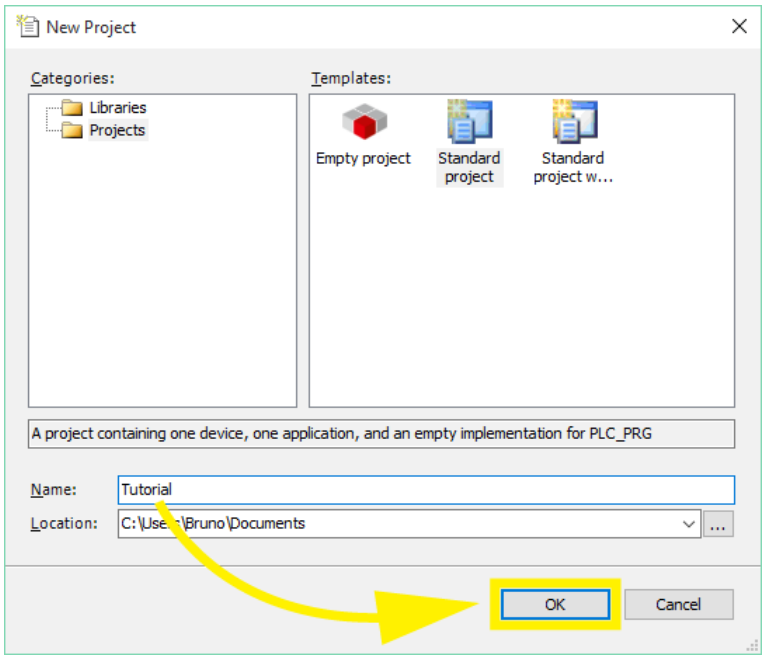


Рисунок 5.6 Вікно створення нового проекту

Слід вибирати ярлик Standart Project зі списку шаблонів та вказати назву проекту (наприклад Tutorial), після чого натиснути ОК.

У вікні Standard Project (рис.5.7) потрібно обрати пристрій керування (Device) CODESYS Control Win V3 (3S - Smart Software Solutions GmbH) та у вікні PLC_PRG мову програмування Structured Text (ST). Натиснути ОК.

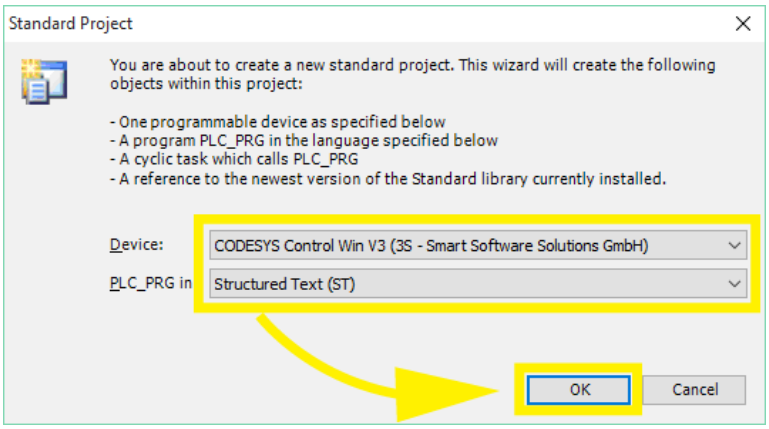


Рис.6. Вікно Standard Project

Натиснути ПКМ на Application (рис. 5.8) і вибирати пункт Add Object > Global variable List... В списку імен увести FIO та натиснути кнопку Add.

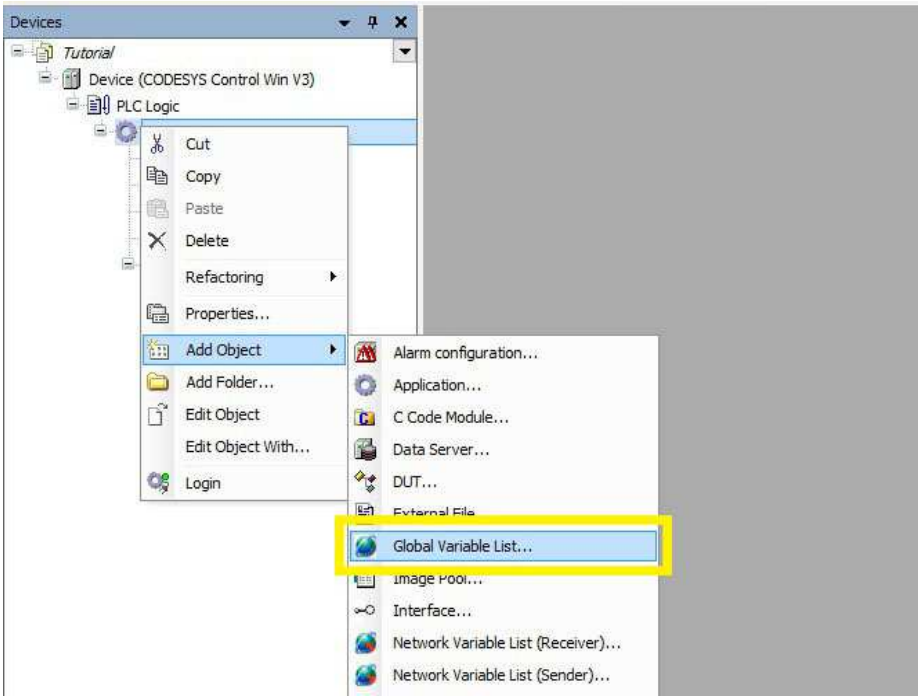


Рисунок 5.8 Випадаюче меню пункту Application

У результаті попередніх дій, повинно автоматично відкритись вікно FIO. Якщо цього не відбулось то це можна зробити за допомогою ЛКМ x2 по пункту FIO.

Далі потрібно прописати глобальні змінні. Ці змінні будуть використовувати FACTORY I/O та CODESYS для обміну даними між собою через OPC Data Access (вони є точками входів / виходів). Для прикладу, що приводиться, вікно FIO буде мати вигляд як на рисунку 5.9.

Наступним кроком буде написання програми на основі складеного алгоритму роботи для заданого варіанту обладнання. Приклад коду програми для сцени з прикладу («Sorting by Height») можна переглянути у Додатку Д.

Після того як програма буде написана , потрібно запустити CODESYS Control Win V3 (рис. 5.10), відповідний ярлик програми можна знайти на робочому столі.

FIO

```
{attribute 'qualified_only'}
VAR_GLOBAL
    iAtLeftEntry:BOOL;
    iAtLeftExit:BOOL;
    iAtRightEntry:BOOL;
    iAtRightExit:BOOL;
    iAuto:BOOL;
    iEmergency_Stop:BOOL;
    iHigh_sensor:BOOL;
    iAtLoaded:BOOL;
    iLow_sensor:BOOL;
    iManual:BOOL;
    iPallet_sensor:BOOL;
    iReset:BOOL;
    iStart:BOOL;
    iStop:BOOL;
    iFactoryReset:bool;

    oConveyorEntry:BOOL;
    oConveyorLeft:BOOL;
    oConveyorRight:BOOL;
    oCounter:INT;
    oEmitter:BOOL;
    oLoad:BOOL;
    oRemoverLeft:BOOL;
    oRemoverRight:BOOL;
    oResetLight:BOOL;
    oStartLight:BOOL;
    oStopLight:BOOL;
    oTransfLeft:BOOL;
    oTransfRight:BOOL;
    oUnload:BOOL;
END_VAR
```

Рисунок 5.9 Вікно FIO з прописаними глобальними змінними

CODESYS Control Win V3

```
stener_Open: Failed to create listen socket!

01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo= ***************
*********************************************
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo=
OPC UA Server
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo=        URL:
opc.tcp://Gordo:4840
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo=        URL:
opc.tcp://192.168.1.101:4840
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo=        URL:
opc.tcp://127.0.0.1:4840
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo= ***************
*********************************************
01458318823495: Cmp=CmpOPCUA, Class=1, Error=0, Info=0, pszInfo= OPC UA server c
ould not be initialized. Server not started.
01458318823497: Cmp=CmpApp, Class=1, Error=0, Info=10, pszInfo= Application [<ap
p>Application</app>] started
01458318823497: Cmp=CM, Class=1, Error=0, Info=34, pszInfo= CODESYS Control read
y
01458318823560: Cmp=CM, Class=2, Error=0, Info=0, pszInfo=!!!! CODESYS Control S
ervice: DEMO mode activated, terminating in approx. 120 minutes.
01458318823631: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router
 <instance>2</instance> address to <address>(0365:0001)</address>
```

Рисунок 5.10 Вікно CODESYS Control Win V3

Далі потрібно повернутись до вікна CODESYS та перевірити програму на помилки. На панелі інструментів слід вибрати на Build > Build (F11). Якщо виникли помилки, то потрібно допрацювати програму та повторити компіляцію. За відсутності помилок слід натиснути на Online > Login (Alt+F8). У вспливаючому вікні потрібно підтвердити свої дії, натиснувши Yes.

Далі треба натиснути на Debug > Start (F5). На цьому налаштування коду програми в CodeSys завершено.

Повернутись до «FACTORY I/O» та до відповідної сцени («Sorting by Height»).

Наступним кроком буде підключення віртуального контролера. Для цього слід натиснути File на панелі інструментів та вибирати Drivers у вспливаючому вікні.

Обрати OPC Client Data Access зі списку драйверів (Driver list), як показано на рисунку 5.11.

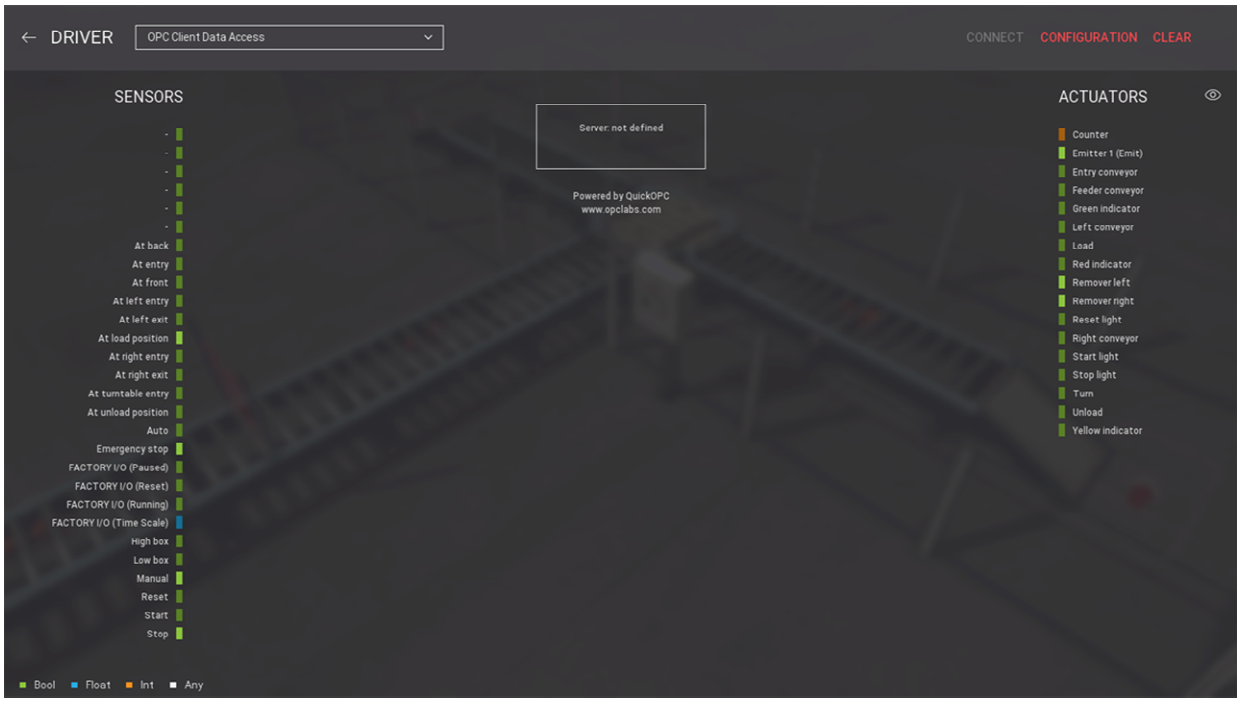


Рисунок 5.11 Вікно драйвера

Натиснути на CONFIGURATION та обрати CoDeSys.OPC.DA зі списку OPC серверів (OPC Server list). Після цього натиснути на BROWSE ITEMS (рис. 5.12).

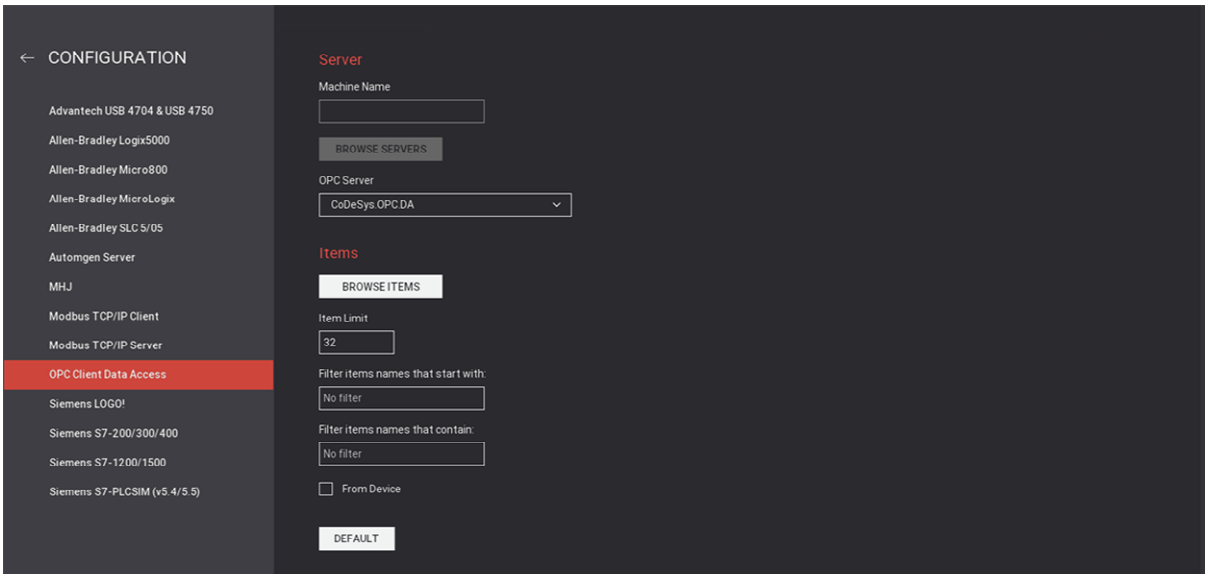


Рисунок 5.12 Вікно CONFIGURATION

Далі потрібно поєднати теги датчиків та виконавчих пристроїв (рис. 5.13) із змінними, які прописувались у програмі.

Рисунок 5.13 Зображення з налаштуваннями віртуального контролера

Після цього потрібно повернутись назад до відповідної сцени та запустити обладнання у роботу (клавіша F5). У випадку некоректної роботи програми необхідно виявити помилки, відкоригувати їх та повторити дії.

Перелік варіантів завдань до даної лабораторної роботи можна переглянути у таблиці 5.5.

Таблиця 5.5 – Варіанти завдань

Варіант	Завдання (Сцена)
1	Sorting Station (Сортувальна станція)
2	Palletizer (Станція палетизатор)
3	Pick & Place (Basic) (Станція для переміщення)
4	Separating Station (Розділювальна станція)
5	Sorting by Weight (Сортування по вазі)
6	Assembler (Станція збирання)

5.4 Сценарій для проведення лабораторної роботи на тему: «Розробка проекту автоматизації у середовищі програмного забезпечення «FACTORY I/O» – CODESYS»

У цій лабораторній роботі необхідно створити сцену згідно варіанту (таблиця 5.6) та вирішити поставлену проблему. Також потрібно розробити програму керування для даної сцени та налагодити її на коректну роботу за допомогою програмного коду на мові ST.

Таблиця 5.6 – Варіанти завдань

Варіант	Завдання
1	Транспортування палет з коробками з другого поверху на перший
2	Транспортування палет з коробками з першого поверху на другий
3	Вкладання поряд двох коробок на одну палету та відвантаження палети
4	Переміщення коробок на палетах, що відрізняються за висотою від заданої на паралельний конвеєр.
5	Перекладання вищих коробок з палети на палету на паралельному конвеєрі
6	Переміщення палет з паралелиних конієрів на один спільний конвеєр

Для наочності буде показано приклад програмування з використанням декількох змінних стану. Нехай система має 2 конвеєри для подачі палет та коробок (рис.5. 14), робота для перестановки коробок, вихідний конвеєр. Система призначена для упаковки коробок по дві штуки на одну палету і видачі їх на зовні. Зрозуміло, що в такій задачі використовувати одну змінну стану не раціонально, так як фактично описуються три незалежних об’єкти:

- вхідний конвеєр на який поступають коробки;
- конвеєр для подачі палет;
- робот, що переставляє коробки.

Отже, будемо використовувати три об’єкти і три незалежних змінні: BoxState, PaletState, RobotState котрі описують відповідні об’єкти. В загальному, кількість об’єктів і відповідність змінних стану буде залежити від кількості матеріальних об’єктів і вимог до програми. Різні об’єкти можуть взаємодіяти між собою визначаючи стан один одного.

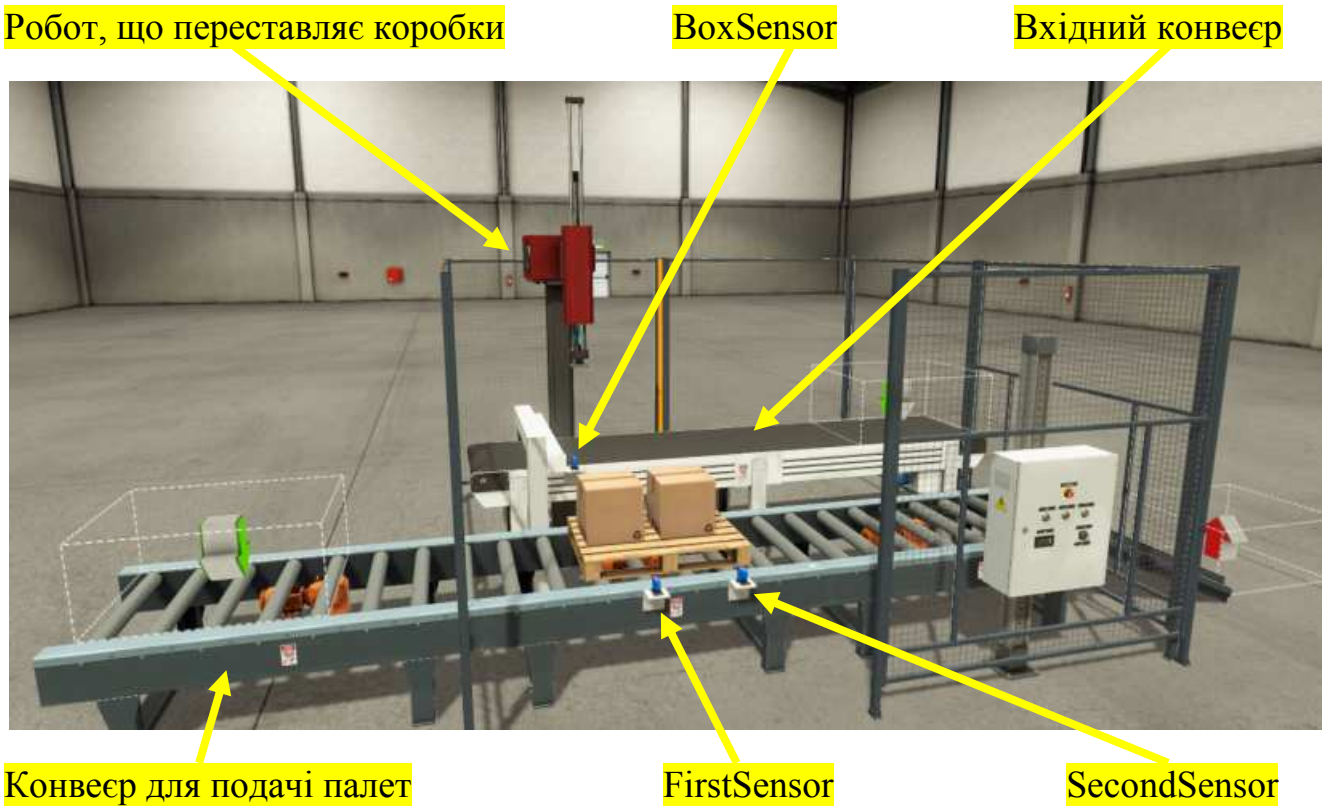


Рисунок 5.14 Приклад сцени у FACTORY I/O

Тепер можна описати відповідні стани об’єктів одночасно вказуючи умови переходів між станами. Лістинг програми можна переглянути у додатку Е.

Вхідний конвеєр та відповідно змінна BoxState може приймати 2 стани:

0) На вивантажувальній позиції відсутня коробка для вивантаження, конвеєр рухається. Умовою виходу з даного стану є спрацювання датчика BoxSensor.

1) Конвеєр зупинений, на вихідній позиції знаходиться коробка. Умовою виходу з даного стану служить стан робота який відповідає переміщенню коробки. Тобто, вхідний конвеєр перейде із 0 в 1 доти, доки робот не забере коробку.

Конвеєр для подачі палет і відповідна змінна PaletState може приймати 5 значень:

0) Немає жодної палети для завантаження, двигуни відповідних конвеєрів включені. Виходом з даного стану буде спрацювання першого датчика позиціонування палети (FirstSensor). Коли датчик спрацює змінна PaletState набуває значення 1.

1) Палета находиться на 1 позиції. Двигун вхідного конвеєра виключений та двигун конвеєра завантажувача вимкнений. Перехід з 1 стану в 2 відбувається тоді коли робот поставив коробку на відповідну палету (і набув свій стан рівний 8).

2) В стані 2 палета рухається на позицію завантаження. Для цього включаються відповідний конвеєр для завантаження. Умовою виходу буде спрацювання 2 датчика (SecondSensor), котрий відповідає 2 позиції завантаження.

3) В 3 стані двигун конвеєра виключається і конвеєр очікує встановлення другої коробки роботом. Умовою виходу з даного стану буде встановлення коробки роботом. Після чого конвеєр для подачі палет переходить у стан вивантаження палет (стан 4).

4) У стані 4 вихідний конвеєр палет включається до тих пір поки датчик 2 позиції не виключиться. Після чого змінна PaletState (і відповідний конвеєр) набуває стан 0.

Робот та відповідна змінна RobotState може приймати 7 станів:

0) В 0 стані захоплюючий пристрій виключений, рука робота втягнута і піднята. Умовою виходу з відповідного стану буде наявність коробки на позиції завантаження та спрацювання датчика наявності палети на першій чи другій позиції завантаження. В такому випадку робот переходить до стану 1.

1) В стані 1 робот здійснює опускання руки до тих пір поки це можливо (і поки рука рухається). Коли рух руки вниз закінчений, робот переходить в стан 2.

2) У стані 2 здійснюється захоплення коробки. Так як захоплюючий пристрій є повільним, то перехід в наступний стан здійснюється після спрацювання таймера. Після витримки часу робот переходить в стан 3.

3) В стані 3 рука робота піднімається вверх. Рука робота піднімається до тих пір поки це можливо. Після закінчення руху руки вверх робот переходить в стан 4.

4) У стані 4 робот витягує руку до спрацювання обмежувача. Коли рука зупиниться здійснюється перехід у стан 5.

5) У стані 5 робот опускає руку з коробкою вниз на відповідну палету. Вихід з даного стану здійснюється коли рука досягнула нижньої позиції. В такому випадку здійснюється перехід у 6 стан.

6) У стані 6 виключається захоплюючий пристрій і через певний час робот переходить у стан 7.

7) У стані 7 робот піднімає руку вверх. Умовою виходу з стану буде неможливість руху руки. Перехід із стану 7 здійснюється у стан 8.

8) У стані 8 робот втягує руку. Умовою виходу є неможливість руху руки. Після закінчення циклу роботи робота він знову повертається у стан 0.

Діаграми станів, що описують зміни станів об’єктів показано на рисунку 5.15.

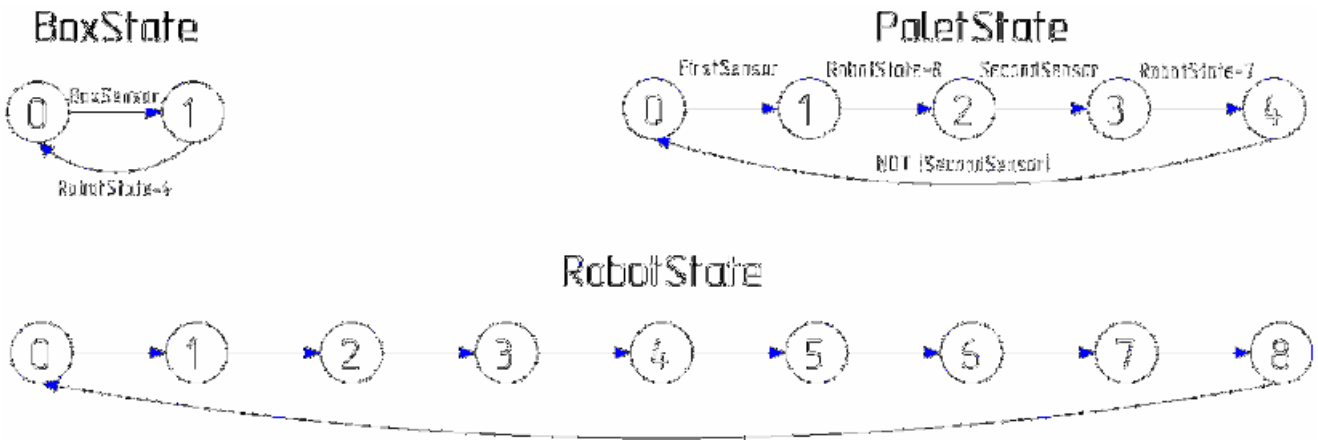


Рисунок 5.15 Діаграма станів до приклад з використанням декількох змінних стану

5.5 Сценарій для проведення лабораторної роботи на тему: «Розробка проекту автоматизації у середовищі програмного забезпечення «FACTORY I/O» за допомогою SoftPLC CONTROL I/O»

У цій лабораторній роботі на основі використання SoftPLC CONTROL I/O разом із «FACTORY I/O» проводиться моделювання роботи автоматизованого обладнання. Слід створити проект автоматизованого обладнняння відповідно до свого варіанту (табл 5.7) та запрограмувати логіку роботи за допомогою SoftPLC CONTROL I/O. Як приклад буде розглядатись сцена - «From A To B (Set and Reset)».

Таблиця 5.7 – Варіанти завдань

Варіант	Завдання (Сцена)
1	Sorting Station (Сортувальна станція)
2	Palletizer (Станція палетизатор)
3	Pick & Place (Basic) (Станція для переміщення)
4	Separating Station (Розділювальна станція)
5	Sorting by Weight (Сортування по вазі)
6	Assembler (Станція збирання)

Необхідно Запустити Factory I/O, пройти по шляху (File / Open / Scenes) та відкрити сцену «From A To B (Set and Reset)» (рис. 5.16).

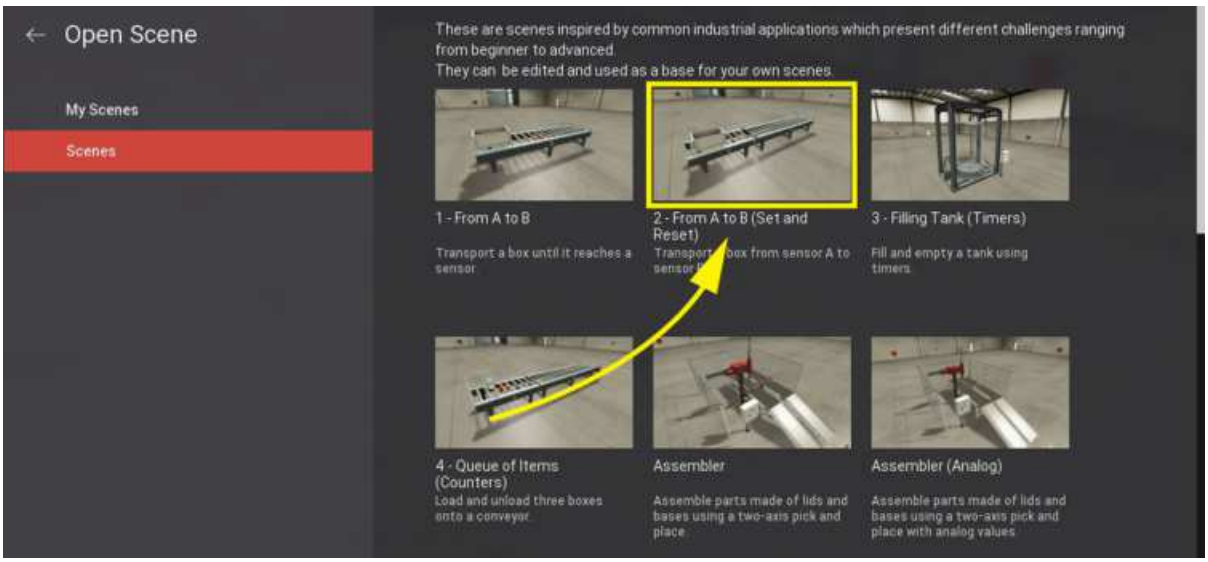


Рисунок 5.16 Вибір сцени From A To B (Set and Reset)

Далі слід перейти по шляху (File / Drivers) або просто натиснувши клавішу F4 та обрати CONTROL I/O зі списку драйверів (рис 5.17). CONTROL I/O запуститься автоматично.

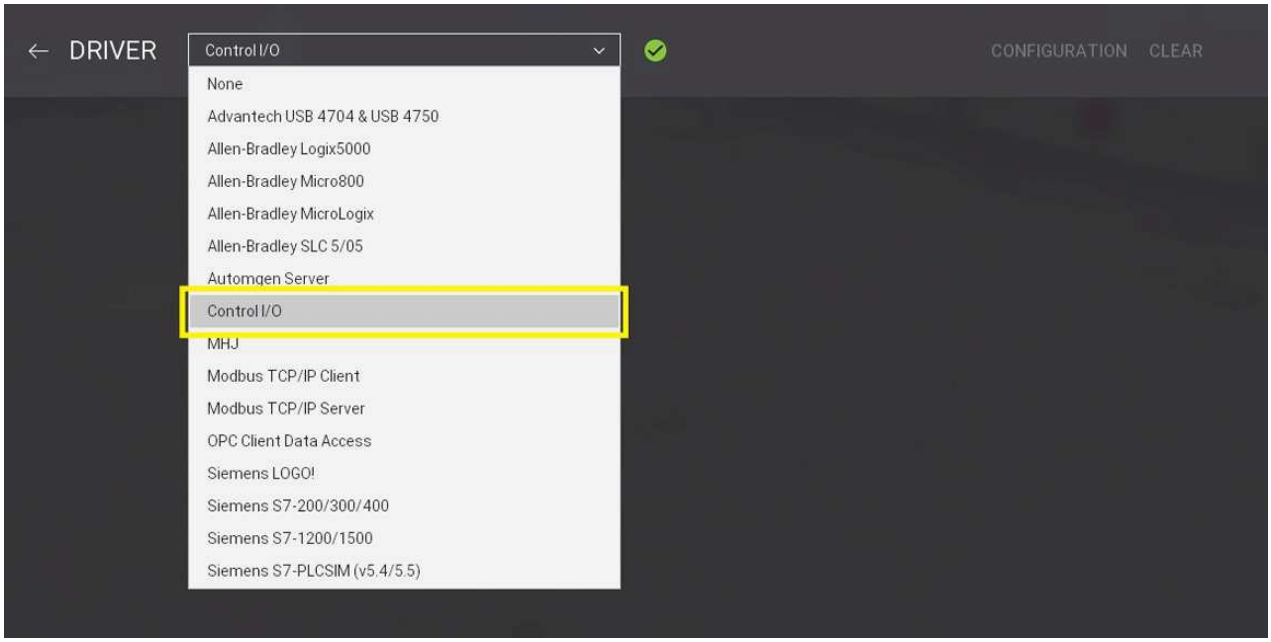


Рисунок 5.17 CONTROL I/O у списку драйверів

Згорнути вікно CONTROL I/O та перейти до відкритої сцени у Factory I/O. Основна задача цієї сцени – транспортувати коробки від Sensor A до Sensor B. Незважаючи на те що це проста сцена, слід врахувати деякі особливості, а саме:

- Entry conveyor (вхідний конвеєр) повинен завжди бути увімкненим (FORCED).
- Sensor A та Sensor B – це світловідбиваючі датчики, які мають нормально замкнені контакти. Тобто, коли нічого не виявлено, значення сигналу на виході датчиків TRUE.

Далі слід переключитись на CONTROL I/O. Якщо використовується один монітор, то можна зробити вікно CONTROL I/O прозорим, клацнувши на кнопці із позначкою «око» на верхній панелі вікна CONTROL I/O (рис. 5.18).

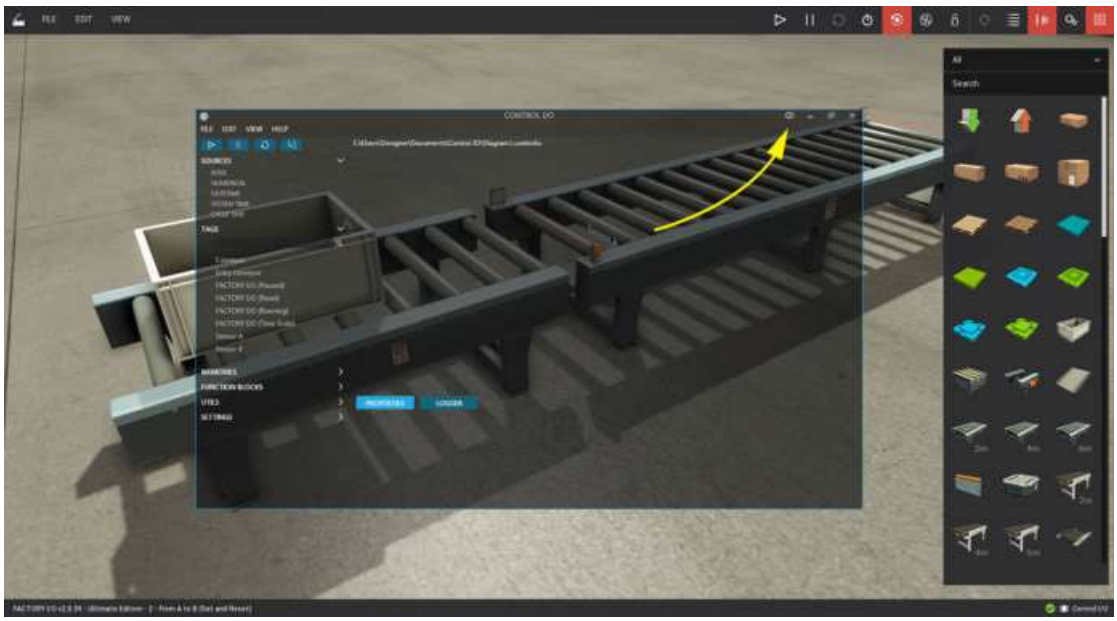


Рисунок 5.18 Увімкнення прозорого режиму в CONTROL I/O

Тепер потрібно встановити з’язки для дій віртуального контролера:

- Після того, як спрацьовує Sensor A повинен увімкнутись Conveyor.
- Після того, як спрацьовує Sensor B, Conveyor повинен вимкнутись.

Теги створеної сцени (Conveyor, Sensor A, Sensor B) можна знайти на панелі TAGS у CONTROL I/O. Тепер потрібно перетягнути теги Sensor A, Sensor B та Conveyor на полотно. Sensor A і Sensor B – це зелені сокети, які представляють собою входи. Conveyor – червоного кольору, який являє собою вихід (рис. 5.19).

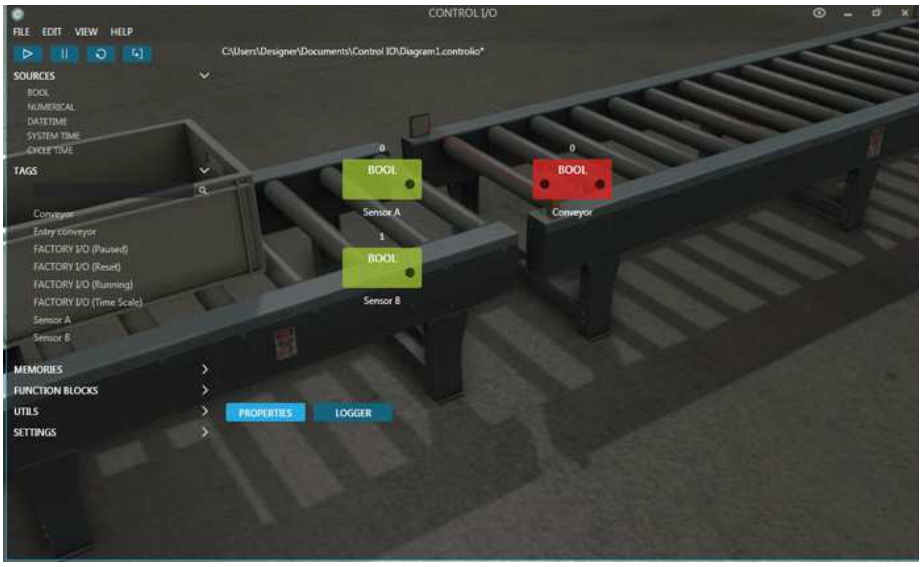


Рисунок 5.19 Сокети Sensor A, Sensor B та Conveyor на полотні

Оскільки Sensor A і Sensor B є датчиками з нормально замкненими контактами, потрібно доповнити їх за допомогою блоків NOT. Відкриваємо вкладку FUNCTION BLOCKS та розгортаємо підвкладку LOGICAL. Тепер перетягуємо блок NOT двічі на полотно та з'єднуємо з сокетами зеленого кольору (рис. 5.20). Тепер, коли Sensor A і Sensor B будуть спрацьовувати, будуть отримуватись значення TRUE.

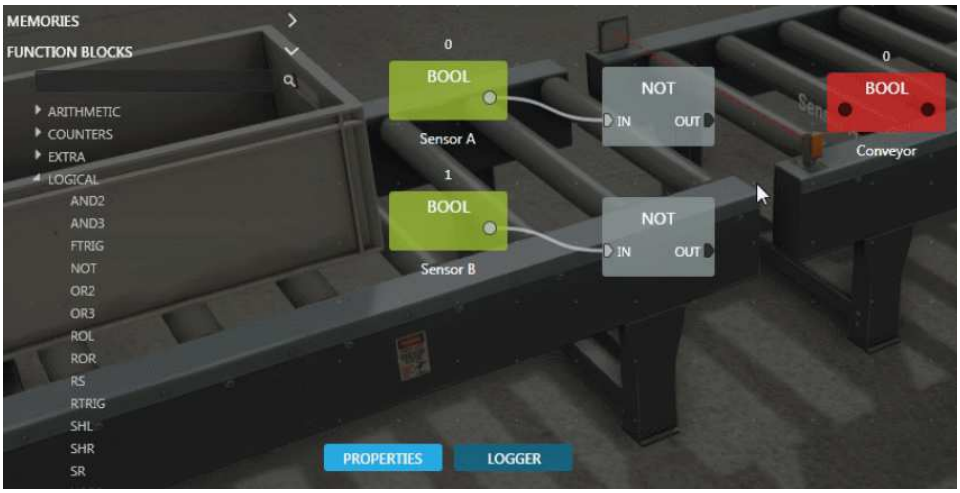


Рисунок 5.20 З'єднані сокети на полотні

Коли Sensor A виявить коробку, це слід запам'ятати. Для цього використовується функціональний блок SR. Цей блок являє собою комірку пам'яті, яку можна встановити та скинути. Потрібно перетягнути блок SR на полотно та з'єднати, як показано на рисунку 5.21.

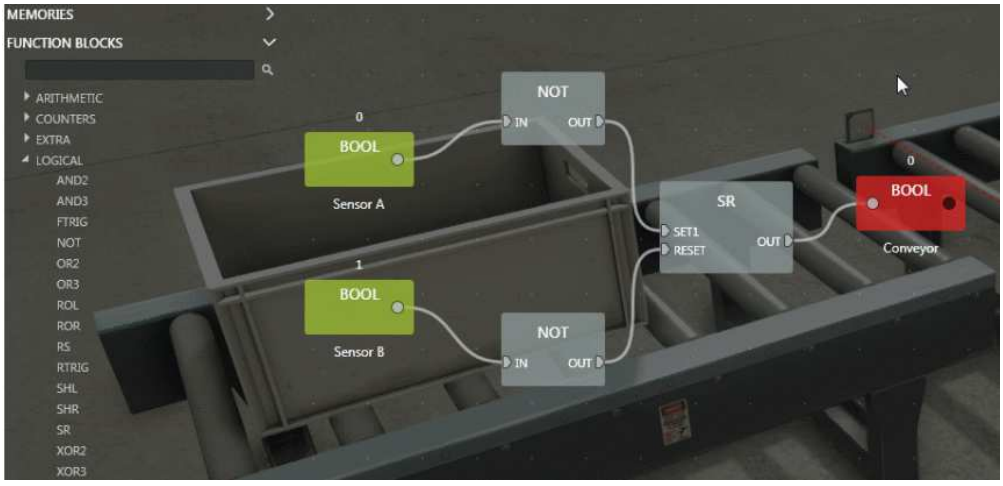


Рисунок 5.21 Готова діаграма з функціональних блоків

Слід зберегти діаграму, вибравши Save в меню File, вказавши потрібний шлях. Тепер можна запустити сцену натиснувши на клавішу RUN (рис. 5.22).

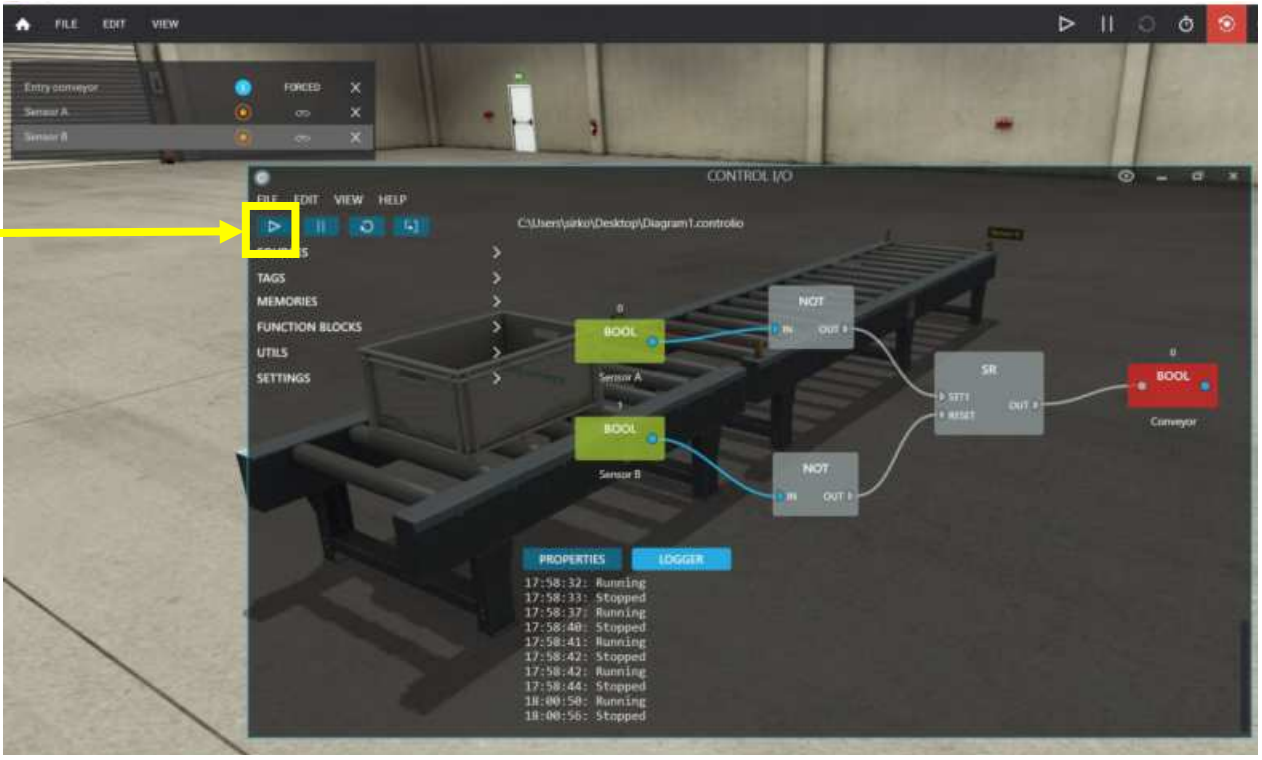


Рисунок 5.22 Запуск сцени в CONTROL I/O

5.6 Висновок

У даному розділі продемонстровано сценарії методичних вказівок до розроблених лабораторних робіт. Крок за кроком показано перебіг л/р, з чого починати, що натискати, куди натискати, які драйвера та компоненти вибирати у суміжних вікнах. Все зроблено для того, щоб у студентів не виникало труднощів при виконанні лабораторних робіт за розробленими методичними вказівками. Майже кожен пункт посилається на скріншот програми, у якому детально показано, які маніпуляції необхідно здійснити на даному кроці.

6 ОБГРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

6.1 Вдосконалення організації наукових досліджень

Основна мета пошукових прикладних науково-дослідних робіт – знаходження нових шляхів дослідження та створення нової техніки та технології виробництва, практичне застосування досліджуваних явищ та фактів, вони направлені на створення нових технологічних процесів, механізмів, машин, виробів, організаційних та економічних структур, методичних рекомендацій На стадії виконання пошукових робіт визначається науково – технічний ефект. Вияснюються можливі області застосування результатів в народному господарстві і на основі комплексного якісного аналізу дається характеристика очікуваної технічної та економічної ваги результатів пошукових робіт. При цьому визначається сукупність показників, які змінюються під впливом впровадження результатів робіт, а також можливий діапазон цих змін.

Основні етапи виконання НДР:

- постановка задачі:
- аналіз досліджень по темі;
- розробка методики та проведення експериментальних досліджень;
- уточнення моделей та оптимізація параметрів та режимів роботи;
- розробка систем керування та інших технічних рішень.

Основними напрямами вдосконалення організації наукових досліджень є:

- розширення області пошуку за рахунок використання нових технічних засобів;
- використання компютерно-інтегрованих технологій, автоматизація процесу вимірювань та скорочення трудоємності і підвищення точності вимірювань;
- розробка подібно-функціональних математичних моделей, реалізація на ПК обчислювального експерименту, що дозволяє уникнути проведення попередніх експериментальних досліджень.

6.2 Планування та розрахунок передвиробничих затрат та капіталовкладень на проведення НДР

При планування затрат на виконання НДР розрізняють передвиробничі затрати $Z_{НДР}$ та капіталовкладення $K_{НДР}$.

Передвиробничі затрати складаються із затрат на виконання таких робіт: постановка задач НДР та розробка технічного завдання; теоретичні дослідження та огляд літератури; лабораторні та заводські дослідження; проектування та конструювання виробів, обладнання, оснастки техпроцесів, цехів і т.д., що є об'єктами НДР; виготовлення, випробування та підналадка зразків. Всі розглянуті затрати є поточними затратами для виконання НДР. Проте при визначені капіталовкладень та госпрозрахункового економічного ефекту від впровадження результатів НДР, виробничі затрати повинні впроваджуватись разом з поточними затратами виробництва нових видів продукції, обладнання і т.д.

Капіталовкладення, які необхідні для виконання НДР, складаються із вкладів в лабораторне обладнання, апарати, прилади з врахуванням затрат на їх проектування та монтаж; в будови та споруди лабораторій, необхідність в яких обумовлена виконанням даної НДР.

Капітальні вклади в НДР складають окремими складовими (додатками) в загальну суму, разом з прямими вкладами в підприємство, що виготовляє продукцію, а також (спряженими і супутніми) вкладами і інші галузі. Їх величина приймається в частині, що відповідає зайнятості обладнання лабораторії, будов та інших засобів на протязі року виконання розглядуваної НДР.

Для визначення передвиробничих (поточних) затрат на виконання НДР складається кошторис затрат, вихідними даними для якого є:

- план проведення НДР;
- розрахунок вартості обладнання для проведення НДР;
- план потреби в основних та допоміжних матеріалах та готових покупних виробах;
- план по праці та заробітній праці.

Для планування праці та заробітної плати на виконання НДР необхідно визначити:

1. Етапи впровадження НДР.
2. Трудомісткість етапів в людино–годинах, людино–днях.
3. Кількість учасників, що виконують роботи по окремих етапах.
4. Тривалість окремих етапів НДР в днях.

Трудомісткість НДР та її окремих етапів визначається за даними НДЧ ТНТУ. На основі трудомісткості встановлюється чисельність робочих, фонд зарплати. Оплата праці науково – технічного персоналу основних лабораторій, проводиться у відповідності їз схемою посадових окладів, затверджених вищестоящою організацією для даного вузу чи НДІ. Посадовий оклад повинен мати вилку.

Після розрахунку кількості робочих та фонду зарплати слід визначити продуктивність праці (відношення кошторисної вартості робочих до кількості працюючих) та середню зарплату.

6.2.1 Розрахунок вартості обладнання для проведення НДР

Вартість обладнання:

$$K_{НДР} = \sum_{i=1}^{n} K_{НДР}^{i} \cdot N_i \cdot \eta_i \text{ , де}$$

n – кількість найменувань обладнання та інших засобів, які застосовуються для виконання НДР;

$K_{НДР}^{i}$ – вартість одиниці засобів i–того виду, $^{грн}/_{од}$;

N_i – кількість екземплярів i–того засобу, необхідних для виконання дослідження, *од*;

η_i – коефіцієнт зайнятості засобу i–того виду на протязі року для виконання даного i–того дослідження.

Якщо обладнання та прилади будуть використовуватись і після завершення НДР, то необхідно визначити величину амортизаційних відрахувань, що припадають на дану НДР. Амортизація за час (період) використання обладнання складає долю затрат, які припадають на дослідницьку роботу, і визначаються так:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}}, \text{ де}$$

$C_Б$ – балансова вартість обладнання, *грн*;

N_A – норма амортизаційних відрахувань в рік, *%*;

$T_{ГОД}$ – річний робочий фонд часу, *год*;

$T_{ФАК}$ – фактичний час роботи обладнання по дослідній темі, *год*.

Оскільки для проведення досліджень використовувався програмний продукт FACTORY I/O то оцінку його вартості можна переглянути у таблиці 6.1

Таблиця 6.1 – Витрати на програмний продукт FACTORY I/O

№	Назва	Кількість	Вартість
1	Ліцензія для FACTORY I/O Modbus & OPC edition	2 шт.	395€
2	Ліцензія для FACTORY I/O Starter Edition	2 шт.	99€
		Сума	988€

Беручи до уваги курс євро на період досліджень, кінцева вартість в гривні буде становити:

988×25.96=25648,48 грн.

6.2.2 Розрахунок витрат на елетроенергію

Час роботи ПК складав безпосередньо 80% ($\psi = 0{,}8$) від часу проведення експеримент $T_{ФАК}$. Середній коефіцієнт навантаження становив $\eta = 0{,}7$.

Відповідно витрати на електроенергію становлять $E_N = \psi\eta p_E N T_{ФАК}$,

де p_E - ціна електроенергії (за квт год) p_E= 0,168 грн.

$$E_N = \psi\eta p_E N T_{ФАК} = 0{,}8 \cdot 0{,}7 \cdot 0{,}168 \cdot 1{,}5 \cdot 52 = 5{,}34 \text{ грн}$$

Витрати електроенергії на освітлення та комп'ютер:

$$E_O = 0{,}168 \cdot 75 = 12{,}6 \text{ грн.}$$

Сумарні витрати на електроенергію:

$$E_\Sigma = E_N + E_O = 5{,}34 + 12{,}6 = 17{,}94 \text{ грн.}$$

6.2.3 Розрахунок витрат на заробітну плату

Для визначення загальної тривалості проведення наукових досліджень доцільно дані витрат часу на виконання окремих стадій звести у таблицю 6.2.

Таблиця 6.2 – Середній час виконання розробки

Номер і назва етапу	Середній час виконання етапу, год.	
	інженер	керівник
Постановка задачі	24	24
Розробка теоретичних матеріалів для лабораторного комплексу	160	12
Розробка методики та проведення есперементальних досліджень	80	9
Уточнення моделей та оптимізація параметроів та режимів роботи	40	12
Розробка систем керування та інших технічних рішень	64	12
Разом	~368	~69

Основна з/п складається із прямої з/п і доплати, яка при укрупнених розрахунках становить 25% – 35% від прямої з/п. При розрахунку з/п кількість робочих днів в місяці приймаємо рівною 25,4 дні / міс, шо відповідає 203,2 год. / міс. Розмір місячних окладів керівника приймаємо 1200 грн. та інженерів — 600 грн.

Пряма з/п визначається наступним чином:

$$ЗП = (O_i - T_i) / 203,2$$

де O_i — розмір місячних окладів 1-х категорій працівників;

T_i — трудомісткість робіт виконаних працівниками /*-х категорій.

Для інженера:

$ЗП = (600 \times 368) / 203,2 = 1\,086,61$ (грн.);

Для керівника:

$ЗП = (1200 \times 69) / 203,2 = 407,48$ (грн.).

Величина доплат визначається наступним чином:

$$ЗП_1 = ЗП \times K_1$$

де K_1 — коефіцієнт доплат (0,25-0,35).

Приймаємо коефіцієнт доплат рівним 0,3:

Для інженера:

$ЗП_1 = 1\,086,61 \times 0,3 = 325,80$ (грн.);

для керівника:

$ЗП_1 = 407,48 \times 0,3 = 122,24$ (грн.);

Основна з/п визначається наступним чином:

$$ЗП_o = ЗП + ЗП_1$$

Для інженера:

$ЗП_o = 1086,61 + 325,80 = 1412,41$ (грн.;

для керівника:

$ЗП_o = 407,48 + 122,24 = 529,72$ (грн..

Величина додаткової з/п визначається наступним чином:

$$ЗП_д = ЗП_o \times K_д$$

Де $K_д$ — коефіцієнт додаткової з/п (0,05-0,1).

Приймаємо коефіцієнт додаткової з/п рівним 0,1, тоді:

для інженера: $ЗП_д = 1412,41 \times 0,1 = 141,24$ (грн.);

для керівника: $ЗП_д = 529,72 \times 0,1 = 52,97$ (грн.).

Витрати, на проведення розробки програмного продукту крім річного фонду заробітної плати включають ще й соціальні нарахування. Нормативи нарахувань на заробітну плату наступні:

- фонд страхування від безробіття – 1,3%;
- пенсійний фонд – 31,8%;
- фонд соціального страхування – 2,9%;
- фонд соціального страхування від нещасних випадків і професійних захворювань 1%.

Всього норматив нарахувань на заробітну плату інженера становить 37%:

$1553,65 \times 0,37 = 574,85$ (грн.),

а для керівника 37 % :

$582,69 \times 0,37 = 215,60$ (грн.).

Таким чином, результати розрахунку заробітної плати та нарахувань на неї зведемо в таблицю 6.3.

Таблиця 6.3 – Зведена відомість витрат на заробітну плату, грн.

	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата	Нарахування на заробітну плату	Всього витрат на заробітну
		Пряма заробітна плата	Доплати	Всього			
1	Інженер	1086,61	325,80	1412,41	141,24	574,85	2128,50
2	Керівник	407,48	122,24	529,72	52,97	215,60	798,29
Всього		1494,09	448,04	1942,13	194,21	790,45	2926,79

Загальновиробничі витрати при укрупнених розрахунках приймаємо на рівні 80 % від суми основної і додаткової з/п інженера, яка була нарахована за роботу по проведенні досліджень. Аналогічно визначаються адміністративні витрати, які доцільно прийняти на рівні 50 % від суми основної і додаткової з/п інженера. Позавиробничі витрати приймаємо на рівні 5 % від виробничої собівартості. Розрахунок поточних витрат зведемо в таблицю 6.4.

Таблиця 6.4 – Калькуляція собівартості проведення НДР

Статті витрат	Витрати, грн.
1. Основна заробітна плата	1942,13
2. Додаткова заробітна плата	194,21
3. Нарахування на заробітну плату	790,45
4. Консультації	50,00
5. Електроенергія	17,94
6. Витрати на програмний продукт FACTORY I/O	25648,48
Повна собівартість	28643,21

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Електробезпека

Електричні установки, з якими доводиться мати справу практично всім працюючим по встановленню та налагодженню засобів автоматизації, виявляють для людини велику потенційну небезпеку, яка збільшується у зв'язку з тим, що органи чуття людини не можуть на відстані виявити присутність електричної напруги на обладнанні.

Степінь ураження електричним струмом залежить від цілого ряду факторів: значення сили струму, електричного опору тіла людини та тривалості протікання через неї струму, виду та частоти струму, індивідуальних властивостей людини та умов навколишнього середовища.

Конструкція електроустановок має відповідати умовам їх експлуатації та забезпечувати захист персоналу від дотику з струмоведучими та рухомими частинами, а обладнання - від попадання всередину посторонніх твердих тіл та води.

У нормальному режимі роботи обладнання – можливість ураження працівників електричним струмом виключена. Але на випадок аварії для запобігання ураження струмом людей передбачене захисне заземлення. Згідно ПУЕ 1.7.65 допустимий опір заземлення повинен бути не більшим 10 Ом.

При виконанні монтажних робіт використовуються переносні електроінструменти (електродрилі, електрошліфувальні установки, тощо). Для забезпечення безпечної праці корпуси однофазних електроприймачів повинні занулюватись.

Захист людини від ураження електричним струмом в мережах з зануленням здійснюється тим, що при замиканні одної з фаз на занулений корпус в ланці цієї фази виникає струм короткого замикання, що діє на струмовий захист (плавкий запобіжник, автомат), в результаті чого відбувається відключення аварійної ділянки від мережі. Крім того, ще до спрацювання захисту струм короткого

викликає перерозподіл напруги в мережі, що приводить до зниження напруги корпусу відносно землі. Таким чином, занулення зменшує напругу дотику та обмежує час, на протязі якого людина, що доторкнулася до корпусу, може попасти під дію напруги.

Для того, щоб забезпечити швидке (на протязі декількох секунд) відключення аварійної ділянки, струм короткого замикання повинен бути достатньо великим. Відповідно до вимог ПУЕ струм короткого замикання повинен не менше ніж в три рази перевищувати номінальний струм плавкої вставки найближчого запобіжника або номінальний струм нерегульованого розчеплювача автоматичного вимикача. При використанні автоматичних вимикачів, що мають тільки електромагнітний розчіплювач (відсічку), струм короткого замикання повинен перевищувати значення струму встановлення миттєвого спрацювання в 1,25-1,4 рази в залежності від номінального струму.

В однофазних електроприймачів, що включені між фазним та нульовим робочим проводами, занулення корпусів слід виконувати з допомогою окремого (третього) провідника, який повинен з'єднувати корпус електроприймача з нульовим захисним проводом. В таких випадках під'єднувати корпуси електроприймачів для забезпечення електробезпеки до нульового робочого проводу недопустимо, оскільки при його розриві (перегоранні запобіжника) всі під'єднані до нього корпуси виявляться під фазною напругою відносно землі.

В мережі з зануленням недопустимо використовувати заземлення окремих електроприймачів, не під'єднавши їх перед цим до нульового захисного провідника. В цьому випадку при замиканні фази на заземлений, але не приєднаний до нульового захисного провідника корпус створюється коло струму через заземлення цього корпусу та заземлення нейтралі джерела струму. Такий випадок небезпечний, оскільки засоби захисту не зможуть відключити такий електроприймач через мале значення струму і тому небезпечна напруга на всіх корпусах може зберігатися тривалий період, поки заземлений приймач не буде відключений вручну.

Важливо відмітити, що якщо занулений корпус одночасно заземлений, то це тільки покращує умови безпеки, оскільки забезпечує додаткове заземлення нульового захисного проводу.

Для ізоляції людини від частин електроустановок, що знаходяться під напругою, використовуються основні та допоміжні ізолюючі засоби, а саме слюсарно-монтажний інструмент з ізольованими ручками, коврики, ізолюючі підставки, тощо.

У приміщеннях, де знаходяться вимірювальні прилади, необхідно забезпечити виконання заходів по боротьбі з статичною електрикою (тобто прилади повинні бути заземлені). Найпростішим засобом є підтримка відносної вологості повітря на рівні 50 - 60 % за допомогою побутового електрозволожувача.

Підлогу слід виконувати відповідно до ГОСТ 12.4.124-83, використовуючи антистатичне покриття на проходах і біля робочих місць.

Оскільки корпуси приладів виконані з металу, то для усунення небезпеки ураження людини електричним струмом (можливий пробій на корпус приладу) використовується захисне заземлення.

7.2 Забезпечення безпеки життєдіяльності при роботі з ПК

Під час роботи на комп'ютерах можуть діяти фізичні та психофізіологічні небезпечні фактори.

Заходи щодо усунення небезпеки ураження електричним струмом зводяться до правильного розміщення устаткування та електричних кабелів. Інші заходи щодо забезпечення електробезпеки, збігаються з загальними заходами пожежо- та електробезпеки.

В якості профілактичних заходів для забезпечення пожежної безпеки слід використовувати скриту електромережу, надійні розетки з пожежобезпечних матеріалів, силові мережі живлення устаткування виконувати кабелями, розрахованими на підключення в 3-5 разів більшого навантаження, включати й

виключати живлення обладнання за допомогою штатних вимикачів. Треба регулярно робити очистку внутрішніх частин комп'ютерів, іншого устаткування від пилу, розташовувати комп'ютери на окремих неспалюваних столах. Для запобігання іскріння необхідно рідше встромляти і виймати штепсельні вилки з розеток.

Система освітлення повинна відповідати таким вимогам:

- освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення - найменшим розміром об'єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об'єкта і фону;

- необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

- на робочій поверхні повинні бути відсутні різкі тіні;

- в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

- величина освітленості повинна бути постійною під час роботи;

- слід обирати оптимальну спрямованість світлового потоку і необхідний склад світла.

Основним обладнанням робочого місця користувача комп'ютера є монітор, системний блок та клавіатура.

Робочі місця мають бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м, між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Робочі місця слід рошташовувати так, щоб уникнути попадання в очі прямого світла. Джерела освітлення рекомендується розташовувати з обох боків

екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, необхідно використовувати антиполискові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах - жалюзі.

Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду. Якщо він розташований під кутом, то стає причиною сутулості. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Перед екраном монітора, особливо старих типів, повинен бути спеціальний захисний екран. При його відсутності треба сидіти на відстані витягнутої руки від монітора.

Фільтри з металевої або нейлонової сітки використовувати не рекомендується, тому що сітка спотворює зображення через інтерференцію світла. Найкращу якість зображення забезпечують скляні поляризаційні фільтри. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

При роботі з текстовою інформацією (в режимі введення даних та редагування тексту, читання з екрану) найбільш фізіологічним правильним є зображення чорних знаків на світлому (чорному) фоні.

Монітор повинен бути розташований на робочому місці так, щоб поверхня

екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Рекомендується розміщувати елементи робочого місця так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Раціональною робочою позою може вважатися таке положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук - вертикальні. Кут ліктєвого суглоба

коливається в межах 70-90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15-20°.

Важливою є форма спинки крісла, яка повинна повторювати форму спини. Висота крісла повинна бути такою, щоб користувач не почував тиску на куприк або стегна. Крісло бажано обладнати бильцями. Його потрібно встановити так, щоб не треба було тягтися до клавіатури. Періодично користувачу необхідно рухатися, вчасно змінювати положення тіла і робити перерви у роботі.

При напруженій роботі за комп'ютером щогодини необхідно робити перерву на 15 хвилин через кожну годину і треба займатися іншою справою. Декілька разів на годину бажано виконувати серію легких вправ для розслаблення.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних принтерах, рекомендується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Вимоги безпеки перед початком роботи на ПК:

- увімкнути систему кондиціювання в приміщенні;
- перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран - під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;
- перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;
- відрегулювати освітленість робочого місця;

- відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

- приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

- ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

- відрегулювати яскравість свічення монітора, мінімальний розмір світної точки, фокусування, контрастність. Не слід робити зображення надто яскравим, щоб не втомлювати очей.

8 ЕКОЛОГІЯ

8.1 Зниження енергоємності та енергозбереження

Енергозбереження спрямоване на зменшення споживання енергії за рахунок використання меншої кількості енергетичних послуг. Енергозбереження відрізняється від енергоефективності, яке стосується використання меншої кількості енергії в тій самій послузі. Наприклад, менше користуватись авто – енергозбереження, а пересісти на авто з меншою витратою палива енергоефективність. Але і енергозбереження, і енергоефективність є техніками зменшення використання енергії.

До заходів пов’язаних з оптимізацією освітлення можна віднести:

- максимальне використання денного світла (збільшення кількості, площі та прозорості вікон);
- оптимальне та раціональне розміщення джерел штучного світла (місцеве, направлене освітлення);
- використання освітлювальних приладів лише за необхідністю;

- підвищення світловіддачі наявних джерел світла (заміна люстр, відбивачів тощо);
- використання приладів управління освітленістю (датчики руху, таймери, датчики освітленості, акустичні датчики, дистанційне керування);
- встановлення інтелектуальних розподілених систем управління освітленням.

До заходів пов’язаних з оптимізацією тепловитрат та підвищенням ефективності систем теплопостачання можна віднести:

- використання сучасного обладнання з вищим ККД тепло генерації (наприклад – конденсаційні котли);
- використання вузлів обліку теплової енергії;
- ізоляція тепломереж для зменшення тепловтрат;
- скорочення шляху від теплоносія до споживача теплової енергії (напр., міні-котельня у будинку)
- оптимізація гідравлічних режимів тепломереж;
- зменшення протікань.
- належна ізоляція опалюваних приміщень;
- використання систем місцевого регулювання опалювальних приладів;
- переведення будинків в режим нульового споживання тепла для опалення (температура всередині підтримується за рахунок внутрішнього тепловиділення та гарної ізоляції).

До заходів пов’язаних з оптимізацією витрат води можна віднести:

- встановлення приладів обліку використання води;
- встановлення зливних бачків, які мають функцію вибору інтенсивності зливу;
- встановлення автоматичних регуляторів витрат води (аератори, сенсорні датчики).

8.2 Джерела електромагнітних полів, іонізуючого випромінення та методи їх знешкодження

Розрізняють природні та штучні джерела електромагнітних полів (ЕМП). У процесі еволюції біосфера постійно перебуває під впливом ЕМП природного походження (природний фон): електричне та магнітне поля Землі, космічні ЕМП (передусім ті, що генеруються Сонцем). У період науково-технічного прогресу людство створило і все ширше використовує штучні джерела ЕМП. У теперішній час ЕМП антропогенного походження значно перевищують природний фон і є тим несприятливим чинником, чий вплив на людину з року в рік зростає. Джерелами, що генерують ЕМП антропогенного походження, є телевізійні та радіотрансляційні станції, установки для радіолокації та радіонавігації, високовольтні лінії електропередач, промислові установки високочастотного нагрівання, пристрої, що забезпечують мобільний та сотовий телефонні зв'язки, антени, трансформатори і т. ін. По суті, джерелами ЕМП можуть бути будь-які елементи електричного кола, через які проходить високочастотний струм. Причому, ЕМП змінюється з тою ж частотою, що й струм, який його створює.

Ще на стадії проектування потрібно забезпечити таке взаємне розташування опромінюючих та опромінюваних об'єктів, яке зводило б до мінімуму інтенсивність опроміненя. Потрібно зменшити імовірність проникненя людей у зони з високою інтенсивністю ЕМП, скоротити час перебування під опроміненям.

Важливе значення мають інженерно-технічні методи захисту: колективний, локальний та індивідуальний. Колективний захист опирається на розрахунок поширення радіохвиль в умовах конкретного рельєфу місцевості. Економічно доцільно використовувати природні екрани – складки місцевості, лісонасадження, нежитлові будівлі. Встановивши антену нагорі, можна зменшити інтенсивність поля, яке опромінює населенний пункт, у багато разів.

При захисті від випромінювання екрана повинне враховуватись затухання хвилі при проходженні через екран (наприклад, через лісову смугу). Для екранування можна використовувати рослинність. Спеціальні екрани у вигляді відбивальних щитів є дорогими та використовуються дуже рідко.

Локальний захист дуже ефективний і використовується часто. Він базується на використанні радіозахистних матеріалів, які забезпечують високе поглинання енергії випромінювання та можуть відбивати її. Для екранування шляхом віддзеркалення використовують металеві листи та сітки з доброю провідністю. Захист приміщеннь від зовнішніх випромінювань можна здійснити завдяки обклеюванню стін металізованими шпалерами, захисту вікон сітками, металізованими шторами. Опромінення у такому приміщенні зводиться до мінімуму, але віддзеркалене від екранів випромінювання, перерозповсюджується в просторі та потрапляє на інші об’єкти.

Іонізуюче випромінювання — це таке випромінювання, взаємодія якого із середовищем призводить до утворення електричних зарядів різних знаків. Розрізняють корпускулярне і фотонне іонізуюче випромінювання.

Джерела іонізуючих випромінювань поділяються на природні та штучні (антропогенні, техногенні).

Штучними джерелами іонізуючих випромінювань є ядерні вибухи, ядерні установки для виробництва енергії, ядерні реактори, прискорювачі заряджених частинок, рентгенівські апарати, засоби зв'язку високої напруги тощо.

Закритими називаються будь-які джерела іонізуючого випромінювання, будова яких виключає проникнення радіоактивних речовин у навколишнє середовище при передбачених умовах їхньої експлуатації і зносу.

Основними принципами забезпечення радіаційної безпеки при роботі із закритими джерелами іонізуючого випромінювання є:

• зменшення потужності джерел до мінімальних значень ("захист кількістю");

• скорочення часу роботи з джерелом ("захист часом");

• збільшення відстані від джерел до людей ("захист відстанню");

• екранування джерел випромінювання матеріалами, що поглинають іонізуюче випромінювання ("захист екраном").

Відкритими називаються такі джерела іонізуючого випромінювання, при використанні яких можливе потрапляння радіоактивних речовин у навколишнє середовище.

При цьому може відбуватися не тільки зовнішнє, але і додаткове внутрішнє опромінення персоналу. Тому, основними принципами забезпечення радіаційної безпеки при взаємодії з відкритими джерелами іонізуючого випромінювання є:

• герметизація виробничого устаткування з метою ізоляції процесів, що можуть стати джерелами надходження радіоактивних речовин у зовнішнє середовище;

• застосування санітарно-технічних засобів та устаткування, використання спеціальних захисних матеріалів;

• використання засобів індивідуального захисту і санітарної обробки персоналу;

• очищення від радіоактивних забруднень поверхонь будівельних конструкцій, апаратури і засобів індивідуального захисту;

ВИСНОВОК

В процесі виконання магістерської роботи було:

- Проведено аналіз наявних програм імітаційного моделювання та вибрано найоптимальніший для розробки лабораторного комплексу;
- Найдено та систематизовано основну інформацію по процедурам розробки автоматизованих систем;
- Розлянуто та наочно продемонстровано основні функціональні можливості програмних продуктів FACTORY I/O та CONTROL I/O;
- Закладено теоретичні та методологічні основи для виконання задач поставлених у лабораторних роботах на основні автоматного програмування;
- Розроблено методичні вказівки до лабораторної роботи на тему: «Ознайомлення з основами роботи у сердовищі програмного забезпечення «FACTORY I/O» та запуск готового проекту»;

- Розроблено методичні вказівки до лабораторної роботи на тему: «Модифікація та відлагодження проекту у середовищі програмного забезпечення «Factory I/O»-CODESYS»
- Розроблено методичні вказівки до лабораторної роботи на тему: «Розробка та відлагодження програми керування технологічним обладнанням у середовищі програмного забезпечення «FACTORY I/O»-CODESYS»
- Розроблено методичні вказівки до лабораторної роботи на тему: «Розробка проекту автоматизації у середовищі програмного забезпечення «FACTORY I/O» – CODESYS»
- Розроблено методичні вказівки до лабораторної роботи на тему: «Розробка проекту автоматизації у середовищі програмного забезпечення «FACTORY I/O» за допомогою SoftPLC CONTROL I/O»

БІБЛІОГРАФІЯ

1. Программируемые логические контроллеры Omron [Електронний ресурс] / TechnoLife, 2019. – Режим доступу: http://t-life.com.ua/catalog/materials/910

2. Программируемый логический контроллер CJ1 OMRON [Електронний ресурс] / Промэнерго Автоматика, 2001 – 2015. – Режим доступу: http://omron-russia.com/plc/cj1m.htm

3. OMRON CS1D [Електронний ресурс] / LKH PRECICON PTE LTD, 2019. – Режим доступу: https://www.precicon.com.sg/product/omron-cs1d

4. Введение в ПЛК: что такое программируемый логический контроллер [Електронний ресурс] / Виктор Чистяков, Терраэлектроника, 2019. – Режим доступу: https://www.terraelectronica.ru/news/5690

5. NEXT-GEN PLC TRAINING 3D FACTORY SIMULATION [Електронний ресурс] / Real Games, 2006 – 2019. – Режим доступу: https://factoryio.com/

6. 3D Simulation Modeling and Analysis Software [Електронний ресурс] / FlexSim Software Products, Inc., 1993 – 2019. – Режим доступу: https://www.flexsim.com/

7. Simulate Robot Aplications [Електронний ресурс] / RoboDK Inc., 2019. – Режим доступу: https://robodk.com/

8. TaraVRbuilder: Creation of dynamic virtual production and logistics systems [Електронний ресурс] / tarakos GmbH, 2000 – 2019. – Режим доступу: https://www.tarakos.de/en/taravrbuilder.html

9. Simcad Process Simulator [Електронний ресурс] / Createasoft, 1992 – 2019. – Режим доступу: https://www.createasoft.com/simulation-software

10. CNCSimulator – Your virtual workshop [Електронний ресурс] / CNCSimulator Inc., 2014 – 2019. – Режим доступу: http://cncsimulator.info/

11. Трегуб В.Г. Проектування систем автоматизації [Текст]: Навч. посібник. – К.: Видавництво Ліра-К, 2016. – 344 с.

12. Пушкар М.С. Проектування систем автоматизації [Текст]: Навч. посібник / М.С. Пушкар, С.М. Проценко – Д.: Національний гірничий університет, 2013. – 268 с.

13. Клепач М.І. Методичні вказівки до виконання курсового проекту з дисципліни «Автоматизація періодичних технологічних процесів» [Текст]: Навч. посібник. – Рівне: НУВГП, 2013. – 22 с.

14. About FACTORY I/O [Електронний ресурс] / Real Games, 2006 – 2019. – Режим доступу: https://docs.factoryio.com/

15. About CONTROL I/O [Електронний ресурс] / Real Games, 2006 – 2019. – Режим доступу: https://docs.factoryio.com/controlio/index.html

16. Я.І. Проць, В.Б. Савків, О.К. Шкодзінський, О.Л. Ляшук. Автоматизація виробничих процесів. [Текст]: Навч. посібник. – Тернопіль: ТНТУ ім. І.Пулюя, 2011. – 344с.

17. Шкодзінський О.К., Пісьціо В.П., Сікора Д.А., Герасимів Ю.О. Методичні вказівки до лабораторної роботи №-21 на тему «Ознайомлення з основами роботи у середовищі програмного забезпечення «Factory I/O» та запуск готового проекту» з курсу «Проектування систем автоматизації» [Текст]: Навч. посібник. – Тернопіль: ТНТУ, 2018 - 18 с.

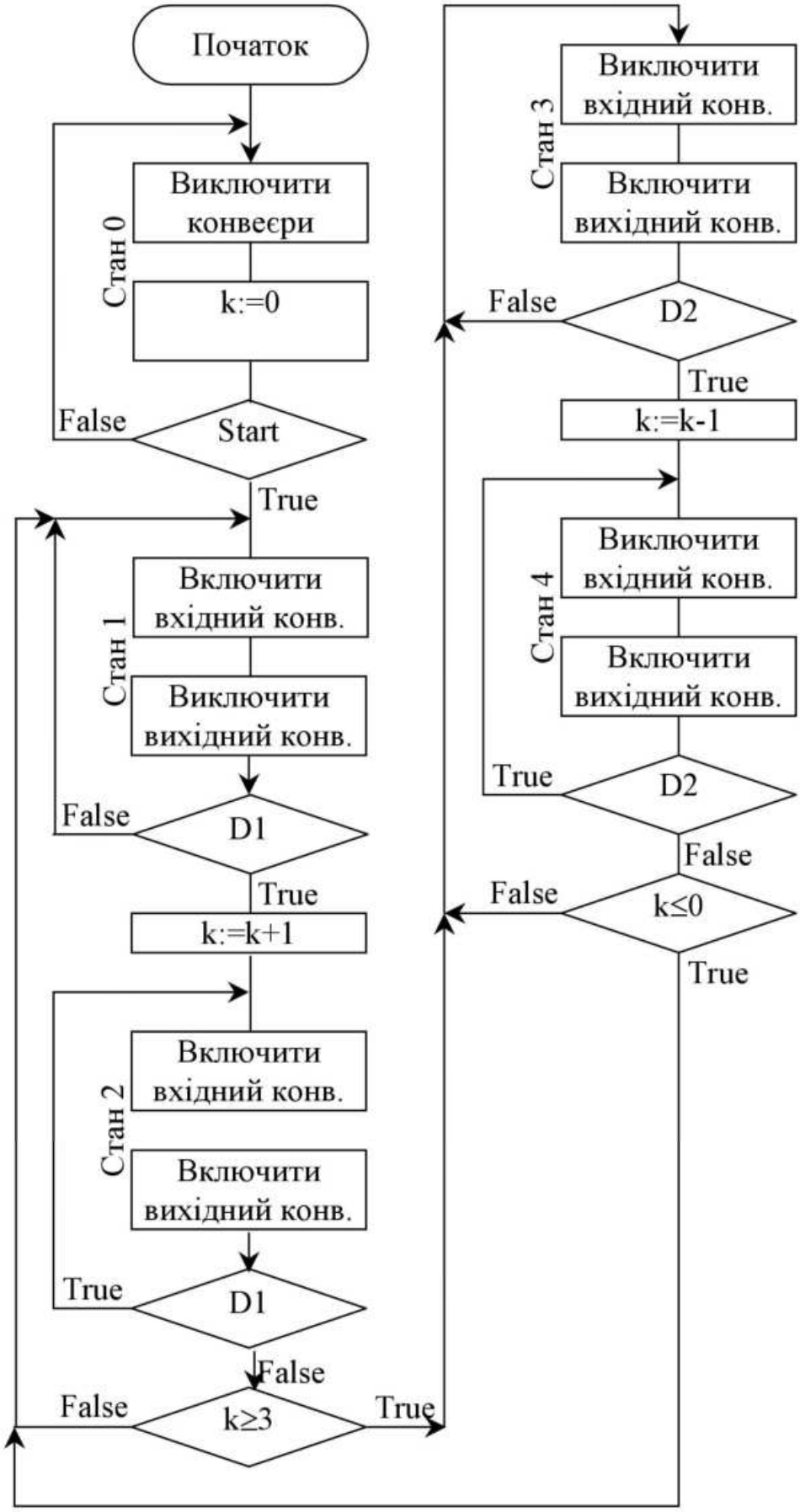
18. Шкодзінський О.К., Пісьціо В.П., Сікора Д.А., Герасимів Ю.О. Методичні вказівки до лабораторної роботи №-22 на тему «Модифікація та відлагодження проекту у середовищі програмного забезпечення «Factory I/O»-COSESYS» з курсу «Проектування систем автоматизації» [Текст]: Навч. посібник. – Тернопіль: ТНТУ, 2018 - 15 с.

19. Шкодзінський О.К., Пісьціо В.П., Сікора Д.А. Методичні вказівки до лабораторної роботи №-23 на тему «Розробка та відлагодження програми керування технологічним обладнанням у середовищі програмного забезпечення «Factory I/O»-CODESYS» з курсу «Проектування систем автоматизації» [Текст]: Навч. посібник. – Тернопіль: ТНТУ, 2018 - 17 с.

20. Шкодзінський О.К., Пісьціо В.П., Сікора Д.А. Методичні вказівки до лабораторної роботи №- 24 на тему «Розробка проекту автоматизації у середовищі програмного забезпечення «Factory I/O» - CODESYS» з курсу «Проектування систем автоматизації» [Текст]: Навч. посібник. – Тернопіль: ТНТУ, 2018 - 22 с.

21. Шкодзінський О.К., Сікора Д.А. Методичні вказівки до лабораторної роботи №-25 на тему «Розробка проекту автоматизації у середовищі програмного забезпечення «Factory I/O» за допомогою SoftPLC CONTROL I/O» з курсу «Проектування систем автоматизації» [Текст]: Навч. посібник. – Тернопіль: ТНТУ, 2019 - 23 с.

Додаток А


Початок
Стан 0
Виключити конвеєри
k:=0
Start
False
True
Стан 1
Включити вхідний конв.
Виключити вихідний конв.
D1
False
True
k:=k+1
Стан 2
Включити вхідний конв.
Включити вихідний конв.
D1
True
False
k≥3
False
True
Стан 3
Виключити вхідний конв.
Включити вихідний конв.
D2
False
True
k:=k-1
Стан 4
Виключити вхідний конв.
Включити вихідний конв.
D2
True
False
k≤0
False
True

```
PROGRAM PLC_PRG;
VAR State:BYTE:=0;
    k:INT;  //Лічильник
END_VAR

CASE state OF
0:  //Початковий стан
    GVL.InpConv:=FALSE;
    GVL.OutConv:=FALSE;
      IF gvl.Start THEN state:=1;
      END_IF
1: //стан 1
    GVL.InpConv:=TRUE; //Включити вхідний конвеєр
    GVL.OutConv:=FALSE;
    IF NOT(GVL.D1) THEN
       k:=k+1;       //Лічильник збільшується при переході із стану 1 у стан 2
       State:=2;
    END_IF;
 2:
     GVL.InpConv:=TRUE;
     GVL.OutConv:=TRUE;
       IF (GVL.D1) THEN
        IF (k<3) THEN
           State:=1;
        ELSE
           state:=3;
       END_IF;
      END_IF;
 3:
    GVL.OutConv:=TRUE;
    GVL.InpConv:=FALSE; //Виключити вхідний конвеєр
      IF NOT(GVL.D2) THEN
      k:=k-1;
      state:=4;
       END_IF;
  4:
    GVL.OutConv:=TRUE;
    GVL.InpConv:=FALSE;
    IF (GVL.D2) THEN
        IF (k>0) THEN  state:=3;
           ELSE state:=1;
         END_IF;
    END_IF;
   ELSE state:=0;
END_CASE
```

```
PROGRAM PLC_PRG
VAR
      State:BYTE:=0;
      h:BOOL;
      Number:INT:=0;
      T1:TON;
END_VAR

T1(IN := FIO.iAtLoadPos , PT:= T#1S);

IF FIO.kReset THEN state:=0;
END_IF

CASE state OF
      0:
      FIO.oEntryConveyor:=TRUE;
      FIO.oLoad := TRUE;
      FIO.oTurn:=FALSE;
      FIO.oUnload:=FALSE;

      IF FIO.iLowBox THEN state:=1;
      END_IF

      1:
      IF  T1.Q
      THEN state:= 2;
      END_IF

      2:
      FIO.oEntryConveyor:=FALSE;
      IF FIO.iAtUnloadPos THEN state:=3;
      IF  Number=0 THEN Number:=1;
      ELSE Number:=0;
      END_IF
      END_IF

      3:
      FIO.oLoad:=FALSE;
      IF NUMBER=0 THEN State:=4;
      ELSE state :=5;
      END_IF

      4:
      FIO.oLoad:=TRUE;
      IF FIO.iAtUnloadPos=FALSE THEN state:=0;
      END_IF

      5:
      FIO.oTurn:=TRUE;
      IF FIO.iLimit90=TRUE THEN state:=6;
      END_IF
```

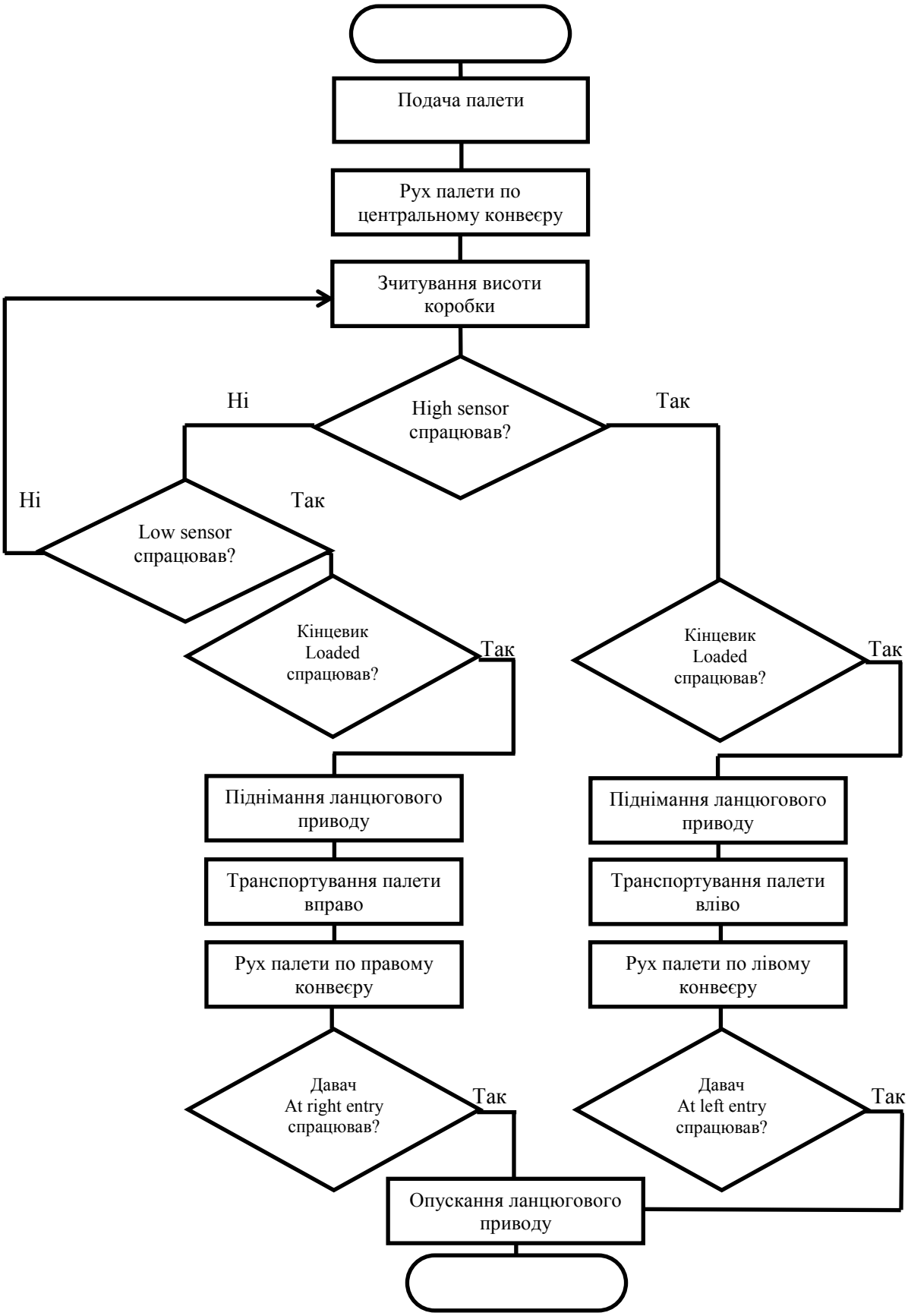
```
        6:
        FIO.oUnload:=TRUE;
        IF FIO.iAtLoadPos=TRUE THEN state:=7;
        END_IF

        7:
        IF FIO.iAtLoadPos=FALSE THEN state:=8;
        END_IF

        8:
        FIO.oTurn:=FALSE;
        FIO.oUnload:=FALSE;
        IF Fio.iLimit0 THEN STATE:=0;
        END_IF

END_CASE
```

Додаток Г


Подача палети
Рух палети по центральному конвеєру
Зчитування висоти коробки
High sensor спрацював?
Ні
Так
Low sensor спрацював?
Ні
Так
Кінцевик Loaded спрацював?
Так
Кінцевик Loaded спрацював?
Так
Піднімання ланцюгового приводу
Транспортування палети вправо
Рух палети по правому конвеєру
Піднімання ланцюгового приводу
Транспортування палети вліво
Рух палети по лівому конвеєру
Давач At right entry спрацював?
Так
Давач At left entry спрацював?
Так
Опускання ланцюгового приводу

Текст програми для сцени «Sorting by Height»

```
PROGRAM PLC_PRG
VAR
      State:INT:=0;
      atState:INT:=1;
      H,L :BOOL;
      stoppressed:BOOL:=FALSE;
END_VAR
```

- Опис змінних

```
IF FIO.iReset THEN
   FIO.oCounter:=0;
   FIO.oCounterL:=0;
   FIO.oCounterR:=0;
END_IF
```

- При натисканні на кнопку Reset всі лічильники будуть скидатись

```
IF FIO.iFactoryReset THEN
   state:=0;
   atstate:=1;
END_IF
```

- Описування дій які будуть відбуватись при натисканні кнопки «Reset the Simulation»

```
IF NOT(fio.iEmergency_Stop) THEN
      IF State<>0 THEN
      atstate:=state;
      state:=0;
      END_IF;
END_IF
```

- Описування дій які будуть відбуватись при натисканні на кнопку екстреної зупинки

```
IF  NOT(fio.iStop) THEN stoppressed:=TRUE;
END_IF;
```

- Якщо кнопка Стоп НЕ натиснена то stoppresed є активним

```
CASE state OF
      0:
         fio.oUnload:=FALSE;
         fio.oConveyorLeft:=FALSE;
         fio.oConveyorRight:=FALSE;
         fio.oConveyorEntry:=FALSE;
         fio.oEmitter:=TRUE;
         fio.oLoad:=FALSE;
         fio.oRemoverLeft:=TRUE;
         fio.oRemoverRight:=TRUE;
         fio.oResetLight:=FALSE;
         fio.oStartLight:=FALSE;
         fio.oStopLight:=TRUE;
         fio.oTransfLeft:=FALSE;
         fio.oTransfRight:=FALSE;
         fio.oUnload:=FALSE;
         IF fio.iStart THEN
            stoppressed:=FALSE;
            IF atstate<>0
            THEN STATE:=atstate;
```

- Опис нульового стану, коли конвеєри виключені

```
   ELSE state:=1;
END_IF; END_IF;
```

```
1:
   fio.oConveyorLeft:=TRUE;
   fio.oConveyorRight:=TRUE;
   fio.oConveyorEntry:=TRUE;
   fio.oLoad:=TRUE;
   fio.oResetLight:=FALSE;
   fio.oStartLight:=TRUE;
   fio.oStopLight:=FALSE;
   fio.oTransfLeft:=FALSE;
   fio.oTransfRight:=FALSE;
   IF FIO.iPallet_sensor THEN STATE:=2;
    END_IF;
    H:=FALSE;
    L:=FALSE;
```

- Увімкнення конвеєрів

```
2:
   fio.oConveyorLeft:=TRUE;
   fio.oConveyorRight:=TRUE;
   fio.oConveyorEntry:=TRUE;
   fio.oLoad:=TRUE;
   fio.oTransfLeft:=FALSE;
   fio.oTransfRight:=FALSE;
   IF FIO.iHigh_sensor THEN H:=TRUE;
    END_IF

   IF FIO.iLow_sensor THEN L:=TRUE;
   END_IF

   IF NOT(FIO.iPallet_sensor) THEN STATE:=3;
    END_IF;
```

- Опис датчиків висоти та датчика поступлення палет

```
3:
   fio.oConveyorLeft:=TRUE;
   fio.oConveyorRight:=TRUE;
   fio.oConveyorEntry:=FALSE;
   fio.oLoad:=TRUE;
    IF fio.iAtLoaded THEN
   fio.oCounter:=fio.oCounter+1;
       IF H THEN state:=4;
        ELSE state:=7;
       END_IF;
    END_IF
```

- Зупинка центрального конвеєра, завантаження палети на транспортер, опис кінцевика (Loaded), вибір напрямку транспортування (лівий чи правий конвеєр)

```
4:
   fio.oTransfLeft:=TRUE;
   fio.oConveyorLeft:=TRUE;
    fio.oLoad:=FALSE;
   IF NOT(fio.iAtLeftEntry) THEN
   fio.oCounterL:=fio.oCounterL+1;
   state:=5;
    END_IF
```

-Рух палети по лівому конвеєру, збільшення лічильника на одиницю

```
5:
   fio.oConveyorLeft:=TRUE;
   fio.oTransfLeft:=TRUE;
    IF fio.iAtLeftEntry THEN state:=6;
   END_IF
```

- Опис давача At Left Entry

```
6:  fio.oTransfLeft:=FALSE;
    state:=10;
```

- Опускання ланцюгового приводу транспортера

```
7:
    fio.oTransfRight:=TRUE;
    fio.oConveyorRight:=TRUE;
    fio.oLoad:=FALSE;
    IF NOT(fio.iAtRightEntry) THEN
    fio.oCounterR:=fio.oCounterR+1;
    state:=8;
    END_IF
```

- Рух палети по правому конвеєру, збільшення лічильника на одиницю

```
8:
    fio.oTransfRight:=TRUE;
    fio.oConveyorRight:=TRUE;
    IF fio.iAtRightEntry THEN state:=9;
    END_IF
```

- Опис давача At Right Entry

```
9:
    fio.oTransfRight:=FALSE;
     state:=10;
```

- Опускання ланцюгового приводу транспортера

```
      10:
           IF stoppressed
           THEN state:=0;
           ELSE state:=1;
           END_IF;
           else state:=0;
END_CASE
```

- Якщо була натиснута кнопка СТОП, то сцена переходить у нульовий стан.

- Якщо нічого не натискалось то сцена переходить у перший стан

```
PROGRAM PLC_PRG
VAR
      BoxState: INT:=0;
      PaletState:INT:=0;
      RobotState:INT:=0;
      GTon:TON;
      GGrabOn:ton;
      TimerOn:BOOL:=FALSE;
      fq: BOOL;
END_VAR

GTon(In := TimerOn,PT:=T#0.5S);
GGrabOn(In:=TimerOn,PT:=T#3S);

CASE BoxState OF
      0:
      G.BoxConv:=TRUE;
      IF G.BoxSensor=TRUE THEN
            G.BoxConv:=FALSE;
            BoxState:=1;
      END_IF
      1:
      G.BoxConv:=FALSE;
      IF RobotState = 4
      THEN
         BoxState:=0;
      END_IF;
END_CASE

CASE RobotState OF
      0: // Скинути робота в 0
      G.MoveX:=FALSE;
      G.MoveY:=FALSE;
      G.Grab:=FALSE;
      TimerOn:=FALSE;
      IF (boxstate =1) AND ((paletState = 3)OR(PaletState=1)) THEN
      robotstate:=1;
      END_IF;

      1: //Опустити руку
      G.MoveY:=TRUE;
      TimerOn:=TRUE;
      IF NOT(G.MovingZ) AND GTOn.Q  THEN
            RobotState:=2;
            TimerOn:=FALSE;
      END_IF

      2://захват
      G.Grab:=TRUE;
      TimerOn:=TRUE;
            IF GGrabOn.Q THEN
            RobotState:=3;
            TimerOn:=FALSE;
```

```
      END_IF

      3://рука вверх
      G.MoveY:=FALSE;
      TimerOn:=TRUE;
      IF NOT(G.MovingZ) AND GTon.Q THEN
            RobotState:=4;
            TimerOn:=FALSE;
      END_IF

      4://руку витягнути
      G.MoveX:=TRUE;
      TimerOn:=TRUE;
      IF (g.MovingX= FALSE) AND GTon.Q THEN
            RobotState:=5;
            TimerOn:=FALSE;
      END_IF

      5://руку вниз
      G.MoveY:=TRUE;
      TimerOn:=TRUE;
      IF (g.MovingZ= FALSE) AND GTon.Q THEN
            RobotState:=6;
            TimerOn:=FALSE;
      END_IF

      6://захоплючий пристій виключити
      G.Grab:=FALSE;
      TimerOn:=TRUE;
      IF Gton.Q THEN
        RobotState:=7;
        TimerOn:=FALSE;
      END_IF

      7://руку вверх
      G.Grab:=FALSE;
      g.MoveY:=FALSE;
      TimerOn:=TRUE;
      IF NOT(g.MovingZ) AND Gton.Q THEN
        RobotState:=7;
        TimerOn:=FALSE;
      END_IF

  8: //руку втянути
      g.MoveX:=FALSE;
      TimerOn:=TRUE;
        IF NOT(g.MovingX) AND Gton.Q THEN
           RobotState:=0;
           TimerOn:=FALSE;
        END_IF;
END_CASE
```

```
CASE PaletState OF
      0://Немає палет для завантаження
      G.StartConv:=TRUE;
      G.SensorConv:=TRUE;
      IF G.FirstSensor THEN PaletState:=1; END_IF;
      1://Палета на 1 позиції
      G.StartConv:=FALSE;
      G.SensorConv:=FALSE;
        IF RobotState=7 THEN PaletState := 2; END_IF;
      2://палета рухається на позицію № 2
      G.StartConv:=FALSE;
      G.SensorConv:=TRUE;
        IF G.SecondSensor THEN PaletState:=3; END_IF;
      3://палета на позиції № 2
      G.StartConv:=FALSE;
      G.SensorConv:=FALSE;
        IF RobotState=6 THEN PaletState := 4; END_IF;
      4://палета готова та переходить на позицію вивантаження
        G.SensorConv:=TRUE;
        IF NOT(G.SecondSensor) THEN PaletState:=0; END_IF;
END_CASE
```