TECHNICAL SCIENCES AND TECHNOLOGIES

DOI: https://doi.org/10.25140/2411-5363-2026-2(44)-100-115
УДК 378.147:621.3:681.51

Олексій Васильович Самойленко

кандидат технічних наук, доцент, доцент кафедри конструювання машин
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
E-mail: o.samoilenko@kpi.ua. **ORCID:** https://orcid.org/0000-0002-0403-1707

ПРОБЛЕМА ПІДГОТОВКИ МАЙБУТНІХ КОНСТРУКТОРІВ-МАШИНОБУДІВНИКІВ ДО ПРОГРАМУВАННЯ СИСТЕМ КЕРУВАННЯ ТЕХНОЛОГІЧНИМ ОБЛАДНАННЯМ

У статті обґрунтовано методику підготовки інженерів-конструкторів до програмування технологічного обладнання на основі програмованих логічних контролерів. Запропоновано цикл із дев'яти занять, що реалізують принцип наскрізного проєктування: від вивчення базового синтаксису до розв'язання комплексного практичного кейсу. Дидактична новизна полягає в паралельному використанні середовищ CODESYS та OpenPLC для опанування популярних мов стандарту IEC 61131-3 (LD, FBD, ST). Результатом є навчальна програма, що забезпечує повний цикл налаштування обладнання – від написання коду до адресації фізичних каналів програмованого логічного контролера. Доведено ефективність виконання ідентичних завдань трьома мовами для формування здатності до вибору оптимального інструментарію в умовах цифрової трансформації виробництва.

Ключові слова: *програмування технологічного обладнання; електротехнічні системи; програмований логічний контролер; мови стандарту IEC 61131-3; середовище розробки; CODESYS; OpenPLC; емулятор; підготовка конструкторів-машинобудівників.*

Рис.: 12. Табл.: 1. Бібл. 13.

Актуальність теми дослідження. Сучасний етап розвитку промисловості характеризується стійкою тенденцією до заміщення складних механічних (а разом з ними гідравлічних та пневматичних) вузлів різноманітними електротехнічними системами. У професійній інженерній спільноті цей процес розглядається як стратегічний напрям модернізації обладнання, що базується на принципі максимальної заміни фізичних компонентів програмно-керованими алгоритмами та засобами мікропроцесорної техніки. Для фахівців машинобудівного профілю це означає докорінну трансформацію об'єкта проєктування: від класичної механічної конструкції до інтегрованого мехатронного комплексу.

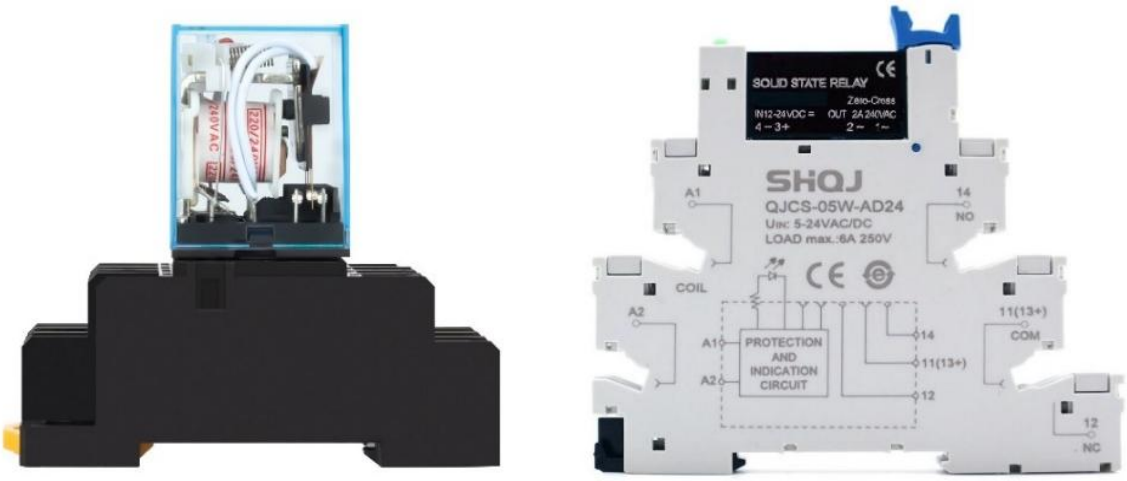
Процес трансформації технічних систем можна простежити на еволюції автомобільних вузлів, де жорсткі механічні зв'язки поступово замінюються керованим електроприводом.

У системах охолодження це проявилося в переході від постійного пасового приводу до використання електромагнітних муфт, що вмикаються за сигналом термореле, а згодом – до повністю автономних електродвигунів під керуванням контролера [3].

У системах газорозподілу шлях пролягав від фіксованої геометрії розподільного валу до механізмів гідравлічної чи електричної корекції фаз («підкручування» вала відносно приводу) [12], а в перспективних розробках – до повного виключення механіки на користь прямого керування клапанами за допомогою електромагнітів [4]. У такому виконанні параметри відкриття клапанів задаються програмним алгоритмом контролера, що дозволяє гнучко підлаштовувати роботу двигуна під різні режими без зміни конструкції вузла.

Аналогічні зміни відбуваються в системах керування та гальмування, де гідропідсилювачі та складні гідравлічні контури поступаються місцем електричним аналогам (EPS та Brake-by-Wire). У таких рішеннях [6] зникає прямий механічний зв'язок між педаллю чи кермом і виконавчим органом: датчики передають дані на контролер, який через програмні алгоритми керує сервоприводами. Це спрощує компонування підкапотного простору, позбавляючи його насосів та магістралей, і дозволяє реалізувати функції стабілізації виключно програмним шляхом.

Сучасна система автоматизації на рівні концепції проєктування прагне до тотального позбавлення механічної складової. Це проявляється не лише в заміні механізмів програмними алгоритмами, а й у глибинній трансформації самої електричної частини. Яскравим прикладом такої еволюції є перехід від електромеханічних пристроїв до твердотільних (рис. 1).



а *б*

Рис. 1. Електромеханічне (а) та твердотільне (б) реле

Джерело: фотографії автора.

Традиційне електромагнітне реле (рис. 1, *а*) за своєю суттю є досить складним механічним вузлом, де рухомий якір та контактна група схильні до зносу, іскріння та інерційності. Натомість твердотільні рішення (рис. 1, *б*) базуються виключно на напівпровідникових компонентах, що дозволяє здійснювати силову комутацію без жодного фізичного переміщення деталей. Відсутність механічного тертя, безшумність та надвисока швидкість реакції роблять такі пристрої ідеальним апаратним фундаментом для сучасної цифрової логіки.

Таким чином, цифровізація витісняє механіку навіть на найнижчому рівні – рівні комутації, що вимагає від майбутнього інженера-конструктора докорінного переосмислення підходів до проєктування обладнання.

Сучасний конструктор проєктує не просто механічний об'єкт, а систему, де алгоритм контролера замінює частину кінематичних вузлів, гідравлічних чи пневматичних систем. Тому необхідність перегляду підходів до підготовки інженерних кадрів зумовлена низкою технологічних та методичних чинників:

– *Спрощення кінематики та цифрова модернізація.* Традиційні багатоступеневі коробки швидкостей, складні системи валів, шестерень і механічних копірів тощо давно й активно заміщуються приводами з прямим цифровим керуванням. Це дозволяє суттєво знизити металоємність обладнання, підвищити його енергоефективність та забезпечити гнучкість переналагодження без втручання в механічну частину.

– *Перехід до програмної логіки керування.* У сфері комутації та автоматики спостерігається стійка тенденція відмови від громіздких електромеханічних релейних схем на користь твердотільних пристроїв та алгоритмів, реалізованих у середовищі програмованих логічних контролерів (ПЛК). Це гарантує вищу надійність, довговічність та швидкість реакції систем у порівнянні з апаратними рішеннями.

– *Пріоритетна електрифікація виконавчих систем.* Попри використання гідравліки (для надвеликих зусиль) чи пневматики (для високих швидкостей) у специфічних нішах, загальним вектором є заміна громіздких магістралей та насосних станцій компактними електроприводами. Така інтеграція не лише спрощує компонування машини, а й дозволяє реалізувати прецизійне керування, недоступне для суто механічних чи пневматичних систем.

– *Нова парадигма інженерних компетенцій.* Оскільки функціональність сучасного обладнання тепер визначається не лише кінематикою, а й алгоритмом роботи контролера, виникає гостра потреба у фахівцях, здатних проектувати механічну систему в нерозривному зв'язку з логікою її керування.

– *Методична адаптація засобами віртуалізації.* Використання середовища розробки у режимі імітаційного моделювання дозволяє продемонструвати процеси цифрової трансформації без залучення вартісного обладнання. Студент (майбутній конструктор) отримує можливість наочно побачити, як програмний код «замінює» фізичний вузол, працюючи у віртуальному середовищі.

Розробка та впровадження методики, що базується на поєднанні проектування в галузі механіки та програмування ПЛК, є критично важливою умовою підготовки конкурентоспроможного інженера для сучасної машинобудівної галузі.

Мета дослідження. Науково обґрунтувати методичну систему підготовки майбутніх конструкторів-машинобудівників до програмування технологічного обладнання, що базується на принципі наступності між апаратно-орієнтованою логікою керування та використанням спеціалізованих мов стандарту IEC 61131-3 для адаптації змісту навчання до реальних вимог сучасного машинобудування.

Результати досліджень. Аналіз навчальних планів підготовки фахівців машинобудівних спеціальностей свідчить, що дисципліни з мікропроцесорного керування технологічним обладнанням зазвичай належать до вибіркової складової, де ресурси та навчальний час виділяються за залишковим принципом. Часто студентам пропонують на вибір два-три паралельні курси, що базуються на різних платформах. Типовим прикладом є конкуренція між вивченням мікроконтролерних систем (наприклад, на базі архітектури Arduino) та опануванням промислових засобів автоматизації.

Незважаючи на те, що цим курсам передує вивчення таких фундаментальних дисциплін, як «Теоретичні основи електротехніки» і «Теорія автоматичного керування» (назви можуть відрізнятись, але суть залишається), у майбутніх конструкторів часто виникає бар'єр при переході від абстрактних розрахунків до практичного алгоритму керування. Коли студент має можливість обрати лише одну з дисциплін, критично важливим стає вибір інструментарію, який забезпечить найбільш швидкий та «безшовний» перехід від механіки до цифрового керування.

У цьому контексті промислово-орієнтований підхід має значну перевагу для майбутнього конструктора. Якщо навчання на базі мікроконтролерів вимагає від студента глибокого занурення у схемотехніку та низькорівневе програмування, то робота з промисловими стандартами дозволяє зосередитися на головному – логіці функціонування розроблюваного технологічного обладнання.

Структура практичної підготовки (рис. 2) відображає логічну трансформацію інженерного мислення студента в процесі навчання. Навчальна траєкторія побудована як послідовний перехід від наочно-фізичного до абстрактно-алгоритмічного проєктування.

Початковий етап – «Ознайомлення із середовищами розробки» – є критичним, оскільки для студента-конструктора професійне програмне забезпечення для автоматизації (IDE) є принципово новим класом інструментів. На відміну від звичних CAD-систем, де робота ведеться передусім з геометрією, середовище розробки систем керування вимагає розуміння логічної структури проєкту, конфігурації входів/виходів та принципів циклічного виконання програми. Це заняття знімає «технологічний бар'єр» перед новим програмним забезпеченням.

Рис. 2. Логіко-структурна схема формування професійних компетентностей у межах комплексу практичних занять

Джерело: рисунок автора.

Логіка побудови *основного навчального блоку* базується на принципі поступового нарощування рівня абстракції програмних моделей. Такий підхід дозволяє нівелювати когнітивний бар'єр при переході від «заліза» до «коду»:

1. *Етап релейної логіки (мова LD)* виступає фундаментальним містком, оскільки візуально та структурно дублює знайомі студентам електротехнічні схеми. Це дозволяє адаптувати наявні знання з ТОЕ до цифрового формату, де замість фізичних контактів студент оперує їхніми програмними образами.

2. *Етап функціональних модулів (мова FBD)* позначає перехід до вищого рівня ієрархії, де керування реалізується через взаємодію готових програмних вузлів – «чорних скриньок». Такий підхід ідеально корелює з сучасним модульним принципом конструювання складних машин.

3. *Етап алгоритмічного програмування (мова ST)* є своєрідною вершиною методичної траєкторії. Опанування текстового структурування алгоритмів готує інженера-машинобудівника до вирішення задач високої складності (обчислення, масиви, нестандартні цикли), де графічні методи стають занадто громіздкими та неефективними.

Завершальним етапом навчання є налаштування апаратної конфігурації, що дозволяє “приземлити” отримані теоретичні знання на реальне “залізо” шляхом інтеграції розробленого програмного забезпечення з фізичними ресурсами контролера. Цей процес передбачає роботу з репозиторієм пристроїв для інсталяції XML-описів конкретних моделей ПЛК, узгодження бібліотек введення-виведення та параметризацію комунікаційних шлюзів для встановлення стабільного зв'язку.

Деталізована структура практичної підготовки (рис. 3) розроблена з урахуванням стислого навчального графіка та специфіки випускового семестру. Ключовим чинником при формуванні навчального плану став часовий ліміт (9 практичних занять) та необхідність синхронізації курсу з етапом дипломного проєктування студентів.

З огляду на це, методика орієнтована на інтенсивне опанування прикладного інструментарію, що дозволяє студентам безпосередньо поєднувати навчання з науково-дослідною роботою, зокрема, і над дипломним проєктом. Кожне заняття розглядається як завершений технологічний модуль: від базового створення проєкту до складних алгоритмів керування. Такий підхід дозволяє студентам безпосередньо використовувати отримані результати та програмні напрацювання при розробці конструкторської або технологічної частини випускових кваліфікаційних робіт.

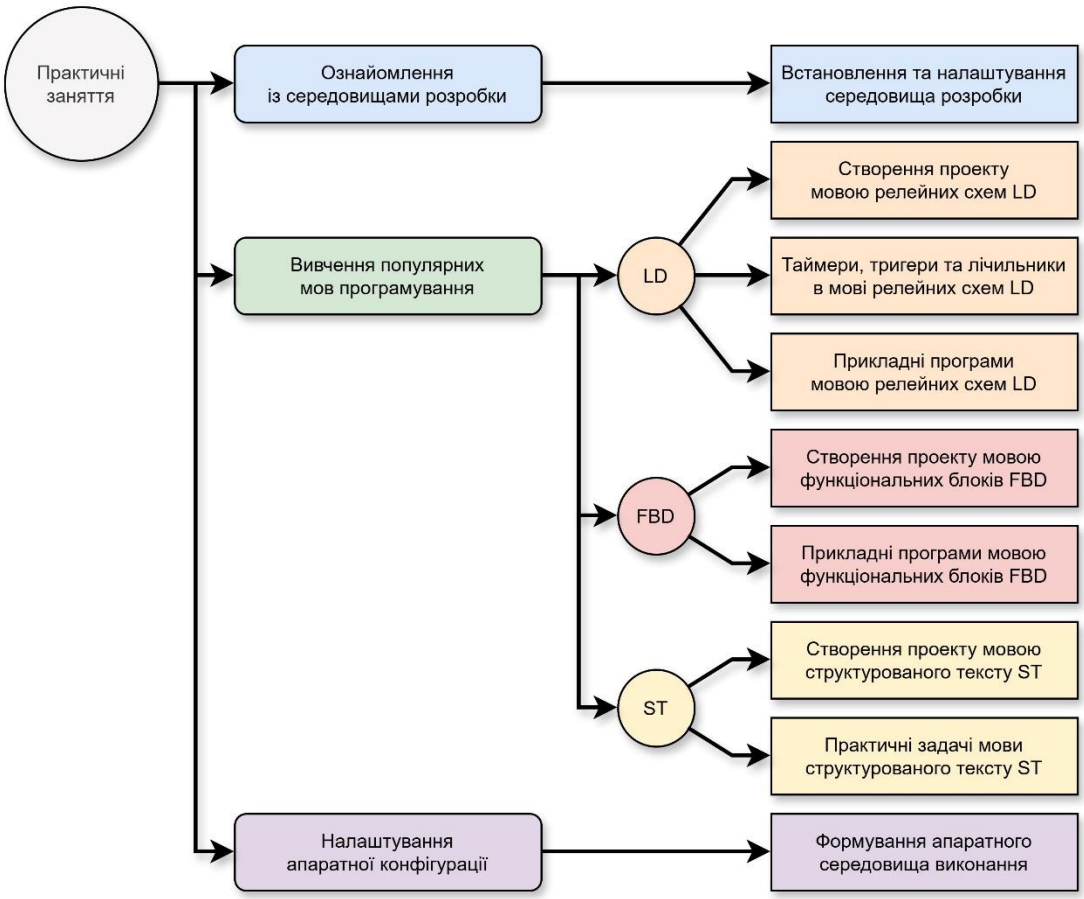


Рис. 3. Розгорнута структура тематичного планування практичних занять
Джерело: рисунок автора.

Для реалізації практичної складової курсу обрано комбінований підхід, що базується на використанні *двох платформ*:

– професійна інженерна екосистема CODESYS Development System V3 [2] (далі по тексту – CODESYS);

– відкрита дидактична платформа OpenPLC Editor [1] (далі по тексту – OpenPLC).

Таке поєднання дозволяє нівелювати високий поріг входження у промислову автоматизацію, забезпечуючи студентам можливість працювати як із світовими стандартами галузі, так і з інтуїтивно зрозумілими інструментами швидкої верифікації.

CODESYS виступає потужним інструментом проєктування, що дозволяє майбутньому інженеру розгорнути повнофункціональний проєкт на власному комп'ютері завдяки лояльній ліцензійній політиці [5] базової версії. На противагу йому, *OpenPLC* слугує ефективним «тренажером», де реалізована можливість візуалізації графіків сигналів за принципом «тут і зараз». Це дозволяє студенту миттєво побачити динаміку процесів, не відволікаючись на складні сервісні налаштування, що є критично важливим на початкових етапах навчання. Порівняльна характеристика середовищ наведена в табл. 1.

Тепер слід розглянути кожне практичне заняття окремо.

Перше практичне заняття є фундаментом курсу, де формується цілісне уявлення про архітектуру сучасної системи керування. Основний акцент зміщено на вивчення «дерева проекту» у середовищі CODESYS, де студент вчиться чітко розмежовувати фізичний рівень (конфігурацію входів/виходів) та алгоритмічний рівень (блоки логіки). Таке занурення дозволяє побачити, як у єдиному цифровому середовищі змінні та типи даних перетворюють розрізнені механічні вузли на керований інтелектуальний об'єкт.

Таблиця 1 – Порівняння середовищ розробки CODESYS Development System V3 та OpenPLC Editor

Характеристика	CODESYS Development System V3	OpenPLC Editor
Тип ПЗ	Промислова інженерна екосистема, світовий стандарт автоматизації	Відкрите ПЗ (Open-source), спеціалізований дидактичний тренажер
Ліцензійна політика	Безкоштовна базова версія	Повністю безкоштовна та відкрита (GNU GPL)
Підтримка мов IEC 61131-3	Повна підтримка всіх 5 стандартних мов + додаткова мова CFC (Continuous Function Chart)	Повна підтримка всіх 5 стандартних мов (LD, FBD, ST, SFC, IL)
Апаратна сумісність (ПЛК)	Підтримка понад 500 брендів. Можливість ручного додавання будь-якого ПЛК через файли опису (EDS, GSD) та бібліотеки виробника	Визначений набір підтримуваних пристроїв. Список розширюється лише розробниками під час оновлення програми; ручне додавання нового ПЛК користувачем обмежене
Сумісність з ОС	Windows	Windows, Linux, Android, macOS
Поріг входження	Високий (через професійну складність інтерфейсу та системних налаштувань)	Низький (лаконічне середовище, орієнтоване на швидкий результат)
Візуалізація	Потужна вбудована система Web- та Target-візуалізації з бібліотеками графічних елементів	Спрощена внутрішня візуалізація, орієнтована на базову верифікацію алгоритмів

Джерело: складено автором.

Під час заняття студенти опановують базовий робочий цикл: від оголошення змінних різних типів до переходу в режим симуляції. Особлива увага приділяється процедурам запуску (Start) та зупинки (Stop) програмного коду, що дозволяє студенту контролювати життєвий цикл алгоритму в реальному часі. Формою звітності за результатами заняття є відповіді на декілька контрольних запитань у вигляді тесту, що дозволяє викладачу оперативно верифікувати засвоєння базових понять інтерфейсу та архітектури проєкту.

Водночас методикою передбачено паралельне використання OpenPLC як початкового майданчика для «м'якого» входження у предметну область. Завдяки лаконічності інтерфейсу та можливості миттєвої візуалізації, студент максимально швидко проходить шлях від створення першої змінної до запуску симуляції. Це дозволяє на практиці відчути реакцію системи на зміну вхідних сигналів, не витрачаючи час на складні системні налаштування, що сприяє швидкій адаптації до принципів логічного програмування.

Друге практичне заняття зосереджене на опануванні принципів побудови логічних схем, де студенти вчаться оперувати програмними «контактами» та «котушками» – прямими аналогами реальних кнопок, датчиків, реле, двигунів тощо . Особлива увага приділяється програмній реалізації базових логічних функцій (AND, OR, NOT, XOR) через послідовне та паралельне з'єднання елементів. Такий підхід дозволяє наочно продемонструвати студенту, як складна апаратна логіка, що раніше потребувала значної кількості фізичних дротів, тепер ефективно реалізується на програмному рівні.

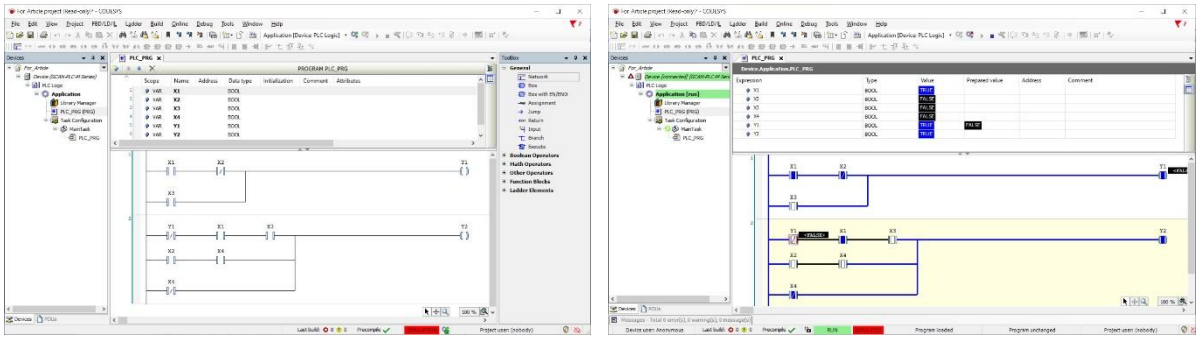
Для закріплення навичок синтезу схем кожному студенту видається індивідуальне завдання у формі системи логічних рівнянь на зразок такої (у цьому випадку – спрощених для демонстрації у статті):

$$\begin{cases} y_1 = (x_1 \wedge \bar{x}_2) \vee x_3 \\ y_2 = (\bar{y}_1 \wedge x_1 \wedge x_3) \vee (x_2 \wedge x_4) \vee \bar{x}_4 \end{cases}.$$

У цих рівняннях змінні x_i відповідають фізичним входам (контактам), а y_j – виходам (котушкам).

Такий підхід забезпечує самостійність виконання та дозволяє охопити широкий спектр логічних комбінацій.

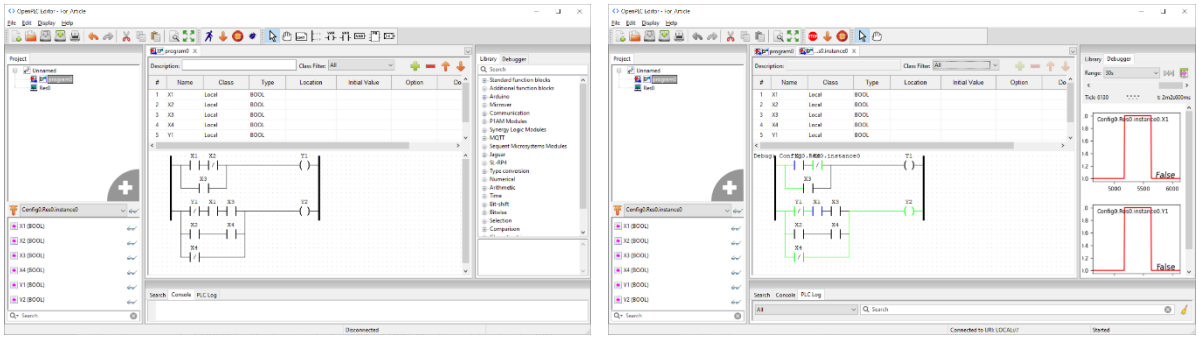
Студенту пропонується реалізувати цю систему рівнянь в середовищах CODESYS (рис. 4) та OpenPLC (рис. 5).



а *б*

Рис. 4. Процес відлагодження індивідуального завдання в середовищі CODESYS: а – реалізація логічної схеми мовою LD; б – верифікація роботи в режимі симуляції

Джерело: розроблено автором.



а *б*

Рис. 5. Процес відлагодження індивідуального завдання в середовищі OpenPLC: а – реалізація логічної схеми мовою LD; б – верифікація роботи в режимі симуляції

Джерело: розроблено автором.

Паралельне використання середовища OpenPLC дозволяє студентам швидше опанувати етап відлагодження завдяки його граничній лаконічності. Попри свою простоту у порівнянні з професійними системами, OpenPLC пропонує надзвичайно ефективну візуалізацію: графічне відображення зміни логічних рівнів у реальному часі (рис. 5, *б*) та інтуїтивно зрозумілий інтерфейс моніторингу змінних роблять процес пошуку логічних помилок максимально наочним. Це дозволяє студенту на початкових етапах зосередитися саме на алгоритмізації, не відволікаючись на складні налаштування промислового середовища.

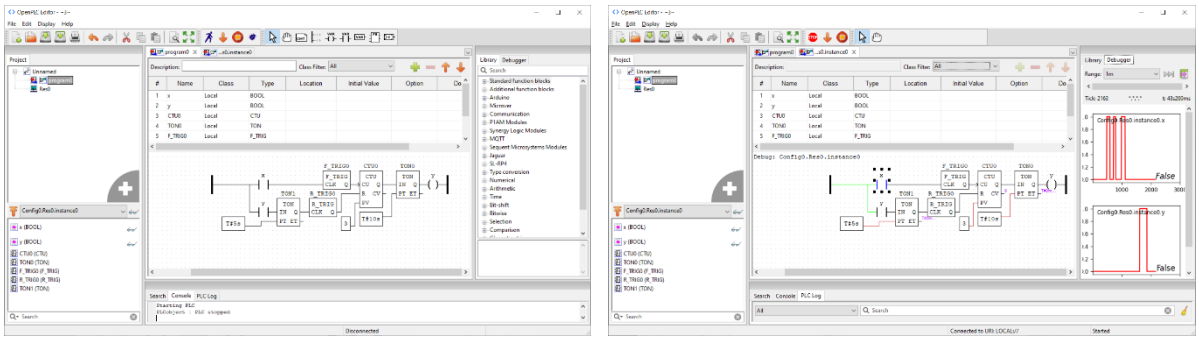
Формою звітності за результатами заняття є: виконані проєкти у форматах обох середовищ розробки; скриншоти коректної роботи програм у режимі симуляції для різних комбінацій вхідних сигналів; відповіді на контрольні запитання, що підтверджують розуміння принципів адресації та взаємодії програмних компонентів.

На *третьому практичному занятті* відбувається перехід до динамічного керування системами. Вивчення таймерів (TP, TON, TOF), тригерів (F- та R-тригери) та лічильників (CTU, CTD, CTUD) дає майбутньому конструктору інструментарій для створення часових затримок, відстеження імпульсних сигналів та підрахунку кількості операцій. Для студента це етап усвідомлення того, що програмний контролер може легко замінити цілі блоки спеціалізованої апаратної автоматики, забезпечуючи при цьому незрівнянно вищу гнучкість налаштувань.

Методика заняття передбачає двоетапний підхід:

1. *Експериментальна частина*: під наглядом викладача студенти випробовують кожен із системних блоків на окремих простих схемах, щоб зафіксувати особливості їхньої поведінки (наприклад, різницю між затримкою увімкнення та вимкнення).

2. *Проєктування комплексної схеми*: студенти реалізують індивідуальне завдання, що інтегрує всі вивчені елементи (контакти, котушки, тригер, лічильник та таймер) у єдиний алгоритм (рис. 6).



а *б*

Рис. 6. Базова комбінована схема керування із застосуванням таймерів, тригерів та лічильників
а – програмна реалізація; б – верифікація роботи в режимі симуляції
Джерело: розроблено автором.

Відмінність індивідуальних завдань на цьому етапі визначається комбінацією типів блоків (наприклад, використання нормально замкнутих контактів у поєднанні з реверсивним лічильником) та заданими константами (кількість імпульсів, тривалість часових інтервалів тощо). Це виключає шаблонність рішень та змушує студента детально аналізувати взаємодію елементів у часі.

Формою звітності за результатами заняття є: проєкти, реалізовані в середовищах CODESYS та OpenPLC; скриншоти роботи програм із демонстрацією стану таймерів та лічильників у режимі симуляції; відповіді на контрольні запитання щодо логіки спрацювання специфічних тригерів та блоків затримки.

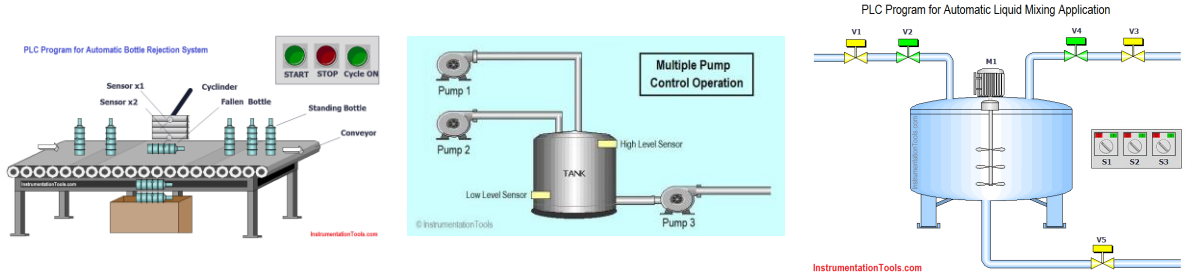
Завершується блок *четвертим практичним заняттям*, де отримані знання інтегруються у вирішення реальних виробничих задач. На занятті спільно з викладачем розбираються три базові прикладні програми, що ілюструють різні типи логіки керування, наприклад:

– система (рис. 7, *а*) відбраковування неправильно орієнтованих предметів (контроль геометрії та положення) [9];

– керування (рис. 7, *б*) декількома насосами з функцією резервування та захисту (реалізація безпекових блокувань та аварійної сигналізації) [7];

– автоматизація процесу (рис. 7, *в*) змішування інгредієнтів у резервуарі (керування послідовністю дій та витримкою часу) [10] тощо.

Після розбору базових кейсів кожен студент отримує індивідуальне завдання за мотивами бази промислових сценаріїв ресурсу Instrumentation Tools [11]. Такий підхід змушує майбутнього конструктора самостійно проходити шлях від аналізу технологічної схеми до розробки робочого алгоритму мовою LD, що стимулює інженерне мислення.



а *б* *в*

Рис. 7. Приклади технологічних об'єктів
для розробки прикладних програм автоматизації мовою LD

Джерело: 1. https://instrumentationtools.com/wp-content/uploads/2018/06/PLC-Program-for-Automatic-Bottle-Rejection-System.png.

2. https://instrumentationtools.com/wp-content/uploads/2019/11/Multiple-Pumps-Control-using-PLC.jpg.

3. https://instrumentationtools.com/wp-content/uploads/2018/06/Automatic-Liquid-Mixing-Application.png.

Формою звітності за результатами заняття є: виконані проєкти у форматах обох середовищ розробки; скриншоти коректної роботи програм у режимі симуляції для різних комбінацій вхідних сигналів.

П'яте практичне заняття фокусується на трансформації інженерного мислення від логіки «цифрової комутації» (дротів) до логіки «потоків даних». Студенти переходять до вивчення мови FBD не «з нуля», а вже маючи досвід розробки прикладних програм мовою LD. Це дозволяє зосередитися не на логіці процесу, яка вже зрозуміла, а на перевагах нової парадигми програмування.

На цьому етапі детально вивчається все різноманіття типів змінних (INT, REAL, TIME тощо) та розширений набір стандартних функціональних блоків. Студенти вчаться групувати повторювані операції у компактні модулі, що значно підвищує читабельність програми. Замість розлогих каскадів реле вони бачать структуровану діаграму, де кожен блок виконує свою функцію: від математичного опрацювання сигналів до реалізації складних регуляторів. Ключовим методичним завданням заняття є реінжиніринг проєкту з попередньої теми. Студентам пропонується реалізувати свій прикладний алгоритм, розроблений раніше мовою LD, засобами мови FBD. Таке порівняння «по гарячих слідах» дозволяє наочно оцінити лаконічність FBD при роботі з аналоговими величинами та обчислювальними операціями.

Формою звітності за результатами заняття є: проекти, реалізовані мовою FBD у середовищах CODESYS та OpenPLC; порівняльні скріншоти ідентичних ділянок коду, виконаних двома мовами (LD та FBD), для демонстрації відмінностей у візуалізації; відповіді на контрольні запитання щодо особливостей передачі даних між блоками та специфіки використання різних типів змінних.

Шосте практичне заняття є кульмінаційним етапом вивчення графічних мов програмування. Його головна мета – навчити студента самостійно проходити шлях від текстового опису технологічної проблеми до створення повноцінного алгоритму керування. На відміну від попередніх етапів, де завдання були чітко структуровані, тут студенти отримують практичний кейс, що містить лише опис об'єкта автоматизації та логіку його функціонування.

Особливістю методики є те, що студент має самостійно:

– визначити необхідну кількість та типи вхідних і вихідних змінних;

– обрати оптимальні функціональні блоки для обробки сигналів та реалізації часових регламентів;

– вибудувати внутрішню логіку взаємозв'язків, що забезпечить безаварійну роботу системи.

Такий підхід стимулює інженерне мислення, адже те саме ТЗ може бути реалізоване різними комбінаціями блоків, що відкриває простір для оптимізації коду. Робота в середовищах CODESYS та OpenPLC на цьому етапі дозволяє студенту не просто «скласти пазл» із блоків, а провести повноцінну пусконалагоджувальну роботу в режимі симуляції, перевіряючи систему на критичні режими роботи.

Приклад індивідуального завдання на шосте заняття (рис. 8):

Завдання

Необхідно реалізувати систему керування насосом для відкачування стічних вод. Насос має вмикатися автоматично, якщо рівень води в резервуарі з урахуванням похибки датчика перевищує критичну позначку, або примусово – натисканням кнопки оператором. Для захисту двигуна від перегріву та частих запусків кожна активація має тривати фіксований час, а загальна кількість запусків повинна фіксуватися для проведення технічного огляду.

Список змінних

Назва	Тип	Початкове значення	Опис
level	INT	700	Початковий рівень води в резервуарі
start	BOOL	FALSE	Кнопка примусового запуску
timer	TP		Таймер роботи насоса
counter	CTU		Лічильник кількості запусків
pump	BOOL	FALSE	Вихід на магнітний пускач насоса
v_counter	INT	0	Поточне значення лічильника
reset	BOOL	FALSE	Кнопка скидання статистики запусків

Логіка роботи програми

- **Математична корекція:** Датчик рівня level має сталу похибку заниження показів. Перед аналізом даних необхідно збільшити отримане значення на величину константи 50.
- **Перевірка умови:** Система повинна сформувати внутрішній сигнал на запуск, якщо скоригований рівень став більшим за критичну позначку 800 одиниць.
- **Керування виходом:** Вихідна змінна pump (насос) має активуватися, якщо спрацювала автоматична умова (п. 2) АБО якщо оператор подав сигнал з кнопки ручного пуску start.
- **Часовий регламент:** Кожна команда на запуск (незалежно від джерела) має вмикати насос рівно на 10 секунд. Протягом цього часу система не повинна реагувати на повторні сигнали або зміну рівня.
- **Статистика та обслуговування:** Програма має підраховувати загальну кількість циклів роботи насоса. Поточне значення зберігається у змінній v_counter. Передбачте можливість обнулення цього значення за допомогою кнопки reset. Максимальна ємність лічильника для перевірки - 100 одиниць.

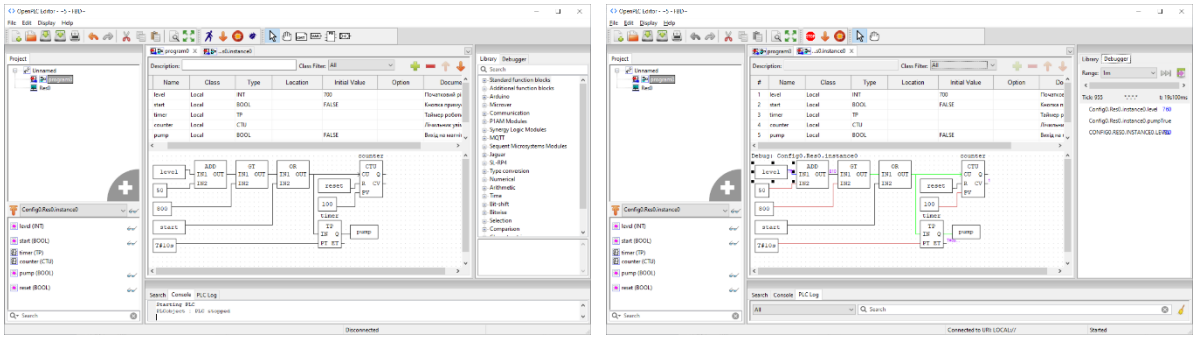
Контрольні показники для тестування

- **Стан спокою:** При level = 700 насос вимкнений.
- **Спрацювання:** При зміні level на 760 (з урахуванням корекції це 810) насос має увімкнутися на 10 секунд.
- **Лічильник:** Після закінчення 10-секундного циклу значення v_counter має збільшитися на 1.
- **Скидання:** Натискання кнопки reset має повернути значення v_counter до 0.

Рис. 8. Приклад індивідуального завдання

Джерело: розроблено автором.

Формою звітності за результатами заняття є: чинний проєкт (рис. 9, *а*), в обох середовищах; демонстрація роботи алгоритму в режимі симуляції (рис. 9, *б*).



а *б*

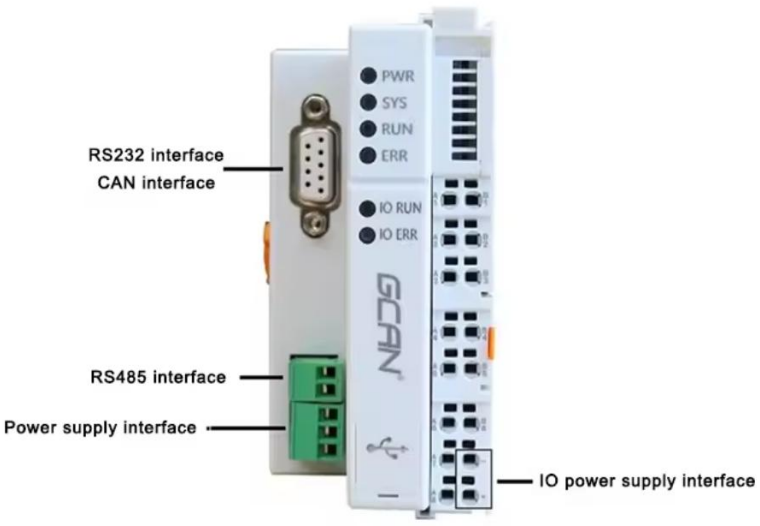
Рис. 9. Варіант реалізації індивідуального завдання (на прикладі середовища OpenPLC)
а – програмна реалізація; б – верифікація роботи в режимі симуляції

Джерело: розроблено автором.

На *сьомому практичному занятті* студенти переходять до вивчення текстової парадигми програмування за стандартом IEC 61131-3. Опанування мови структурованого тексту (ST) починається з вивчення суворого синтаксису: правил оголошення змінних, операторів присвоєння та базових логічних конструкцій (IF-THEN-ELSE, CASE). На цьому етапі студенти виконують завдання, ідентичне за складністю тому, що було реалізоване графічною мовою FBD на четвертому практичному занятті. Це дозволяє наочно порівняти візуальне з'єднання функціональних блоків із написанням текстового коду, акцентуючи увагу на пріоритетності операцій та точності написання інструкцій. Формою звітності є діючий проєкт у двох середовищах (CODESYS та OpenPLC) та відповіді на контрольні запитання.

Восьме практичне заняття присвячене реалізації «Практичного кейсу» мовою ST, що завершує цикл порівняльного аналізу основних мов програмування ПЛК. Ключовим методичним аспектом тут є наскрізне проєктування: студенти реалізують той самий прикладний алгоритм, який вони вже розробляли мовами LD (четверте заняття) та FBD (п'яте заняття). Такий підхід дає студентам унікальну можливість оцінити переваги ST при роботі зі складними математичними обчисленнями та циклами (FOR, WHILE), де графічні мови стають занадто громіздкими. Завершуючи це заняття, студенти роблять висновки щодо вибору інструменту під конкретне інженерне завдання, формуючи критичне мислення конструктора систем автоматизації. Звітність за це заняття аналогічна попереднім етапам і включає демонстрацію роботи алгоритму в режимі симуляції.

Завершальне *дев'яте практичне заняття* є «моментом істини» навчального курсу, де абстрактні алгоритми, відпрацьовані у віртуальних середовищах, переносяться на реальне «залізо». На відміну від OpenPLC, де вибір цільових платформ обмежений заздалегідь визначеним списком, середовище CODESYS дозволяє самостійно конфігурувати параметри практично будь-якого промислового пристрою. Тому завдання за цим заняттям виконується лише в середовищі CODESYS. Головною метою є опанування процедури фізичної конфігурації програмованого логічного контролера (на прикладі моделі GCAN-PLC-302) (рис. 10) [8] та встановлення стабільного зв'язку між інженерною станцією та об'єктом керування.



Rated power supply	24V DC
CAN channel (channel)	1
485 channel (road)	1
232 channel (road)	1
Digital input (channel) P/N type optional	8
Digital output (channel) P/N type optional	6
High-speed counting input	4 (reused with A1/2/3/4)
High-speed pulse output	The first 4 channels (reused with B1/2/3/4)
Expandable number of IO modules	16 pieces
CANopen master/slave protocol	Support
Modbus RTU master/slave protocol	Support
range of working temperature	-40C~+85° C
Protection level	IP20
EMC anti-interference level	Zone B

Рис. 10. Програмований логічний контролер GCAN-PLC-302: загальний вигляд (ліворуч); основні характеристики (праворуч)

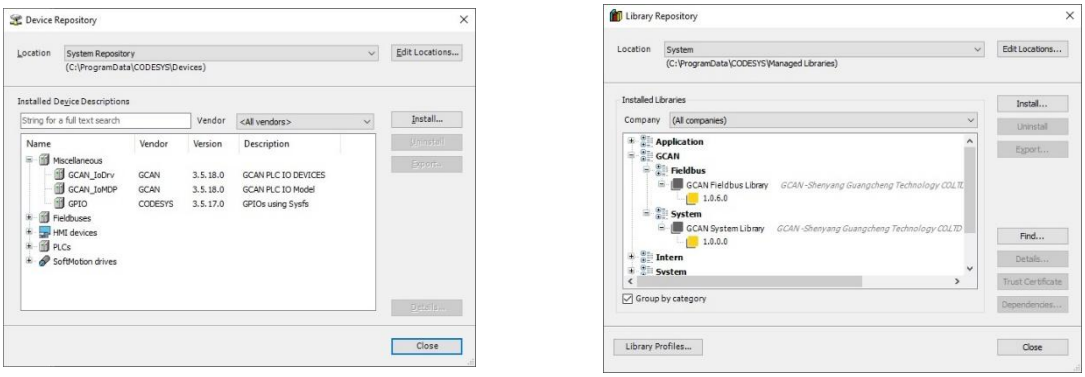
Джерело: https://gcanbus.com/gcan-plc-301-plc/.

На цьому етапі студенти переходять від дидактичних спрощень до виконання суворих регламентів промислової інсталяції:

– *Системна ідентифікація пристрою*: Використовуючи репозиторій пристроїв (Device Repository), студенти інсталюють файли опису специфічної серії ПЛК (рис. 11, *а*). Це дозволяє середовищу CODESYS розпізнати апаратну архітектуру конкретної моделі.

– *Бібліотечна підтримка*: Через менеджера бібліотек (Library Repository) додаються необхідні програмні компоненти та драйвери виробника (рис. 11, *б*), що забезпечують коректну роботу системних функцій та комунікаційних протоколів.

– *Мережева конфігурація та адресація (mapping) фізичних каналів*: Студенти самостійно налаштовують параметри TCP/IP з'єднання, проводять сканування мережі для пошуку активного контролера та виконують прив'язку (mapping) програмних змінних до фізичних адрес вбудованих входів і виходів (%IX та %QX).



а *б*

Рис. 11. Етапи апаратної конфігурації ПЛК GCAN у середовищі CODESYS: а – інсталяція описів пристроїв; б – підключення бібліотек

Джерело: розроблено автором.

На рис. 12 продемонстровано процес зіставлення програмних змінних із фізичними адресами входів та виходів контролера, що є проміжним етапом створення цифрового містка між алгоритмом керування та апаратною частиною системи

Результатом заняття є повністю сконфігурований апаратно-програмний комплекс, де кожен віртуальний об'єкт програми чітко зіставлений із фізичним каналом ПЛК. Це створює необхідні передумови для подальшого впровадження алгоритмів керування в реальні технологічні процеси. Таким чином замикається повний цикл підготовки студента-конструктора: від абстрактного логічного рівня «дерева проєкту» до фінального налаштування промислової апаратної конфігурації, яка є повністю готовою до експлуатації та інтеграції з виконавчими механізмами в умовах виробництва.

Узагальнюючи досвід впровадження описаної методики, слід зазначити, що логічна структура практичної частини курсу (рис. 13) базується на принципі когнітивного ускладнення та наскрізного проєктування. Якщо на початкових етапах (заняття 1…3) переважають атомарні індивідуальні вправи для освоєння базового синтаксису мов LD та FBD, то, починаючи з четвертого заняття, фокус зміщується на реалізацію комплексного «Практичного кейсу». Такий підхід дозволяє студентам провести глибокий порівняльний аналіз ефективності різних мов програмування (LD, FBD, ST) на прикладі одного й того самого технологічного процесу. Це не лише закріплює технічні навички, а й формує у майбутніх інженерів здатність обґрунтовано обирати інструментарій розробки залежно від специфіки задачі автоматизації, що є важливим для їхньої подальшої професійної діяльності.

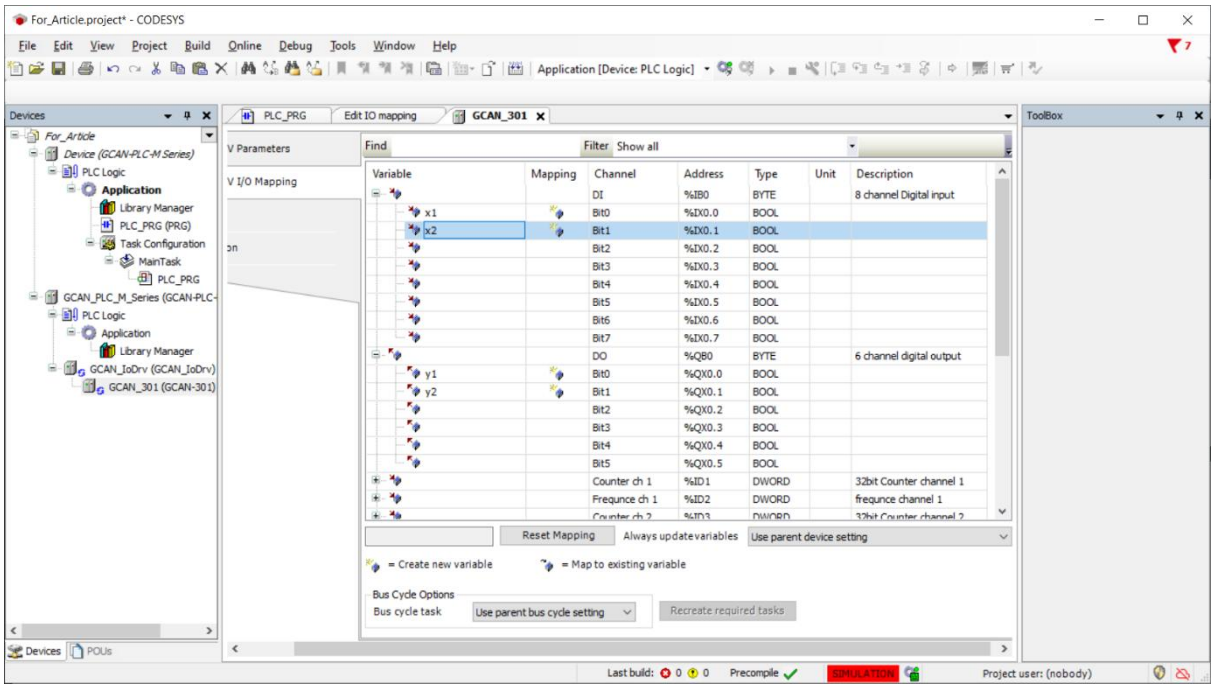


Рис. 12. Етап конфігурування адресного простору та прив'язки змінних (I/O Mapping) для вбудованих модулів ПЛК GCAN

Джерело: розроблено автором.

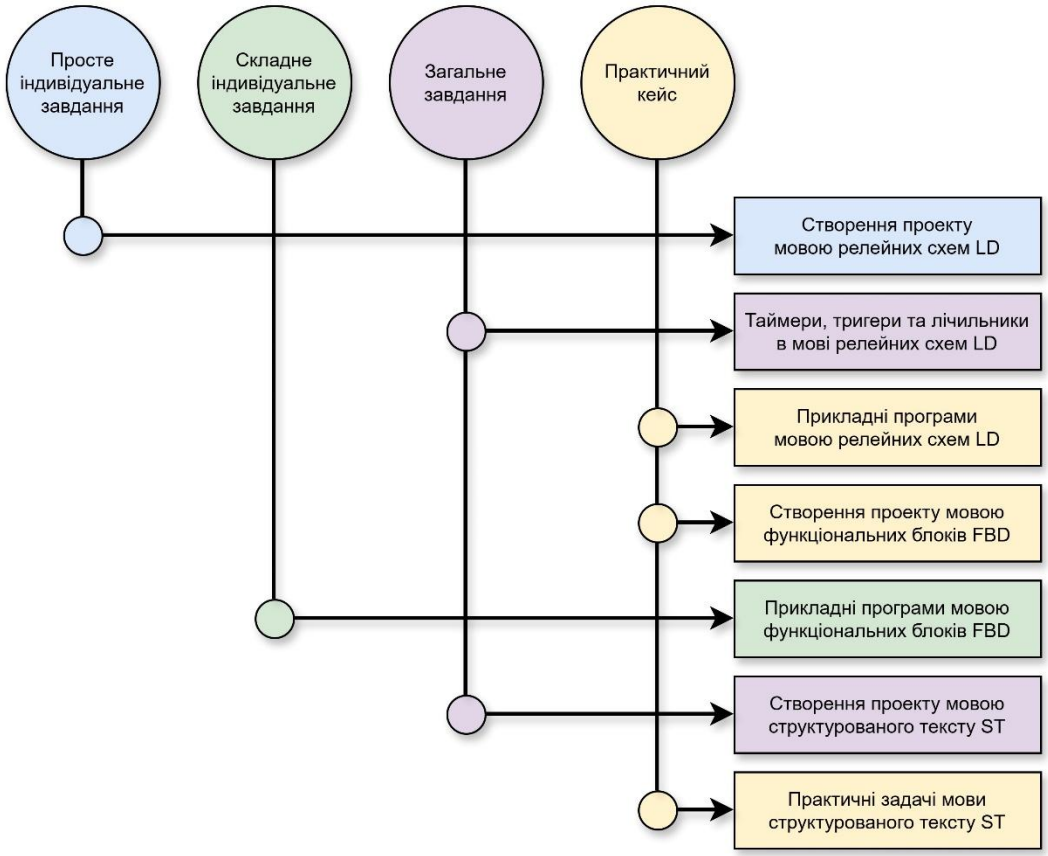


Рис. 13. Матриця розподілу навчальних завдань за рівнем складності та мовами програмування

Джерело: рисунок автора.

Висновки. У процесі дослідження науково обґрунтовано та практично реалізовано методичну систему підготовки майбутніх конструкторів-машинобудівників до програмування систем керування технологічним обладнанням, що відповідає сучасним викликам цифрової трансформації промисловості. Запропонований підхід базується на концепції тотальної цифровізації, де програмний алгоритм розглядається як повноцінна заміна складних кінематичних та механічних вузлів.

Логічна структура курсу, що охоплює дев'ять практичних занять, побудована за принципом поступового нарощування складності. Це дозволило реалізувати наскрізне проєктування: від освоєння базової релейно-контактної логіки (LD) та функціональних блоків (FBD) до написання високоефективного текстового коду (ST). Використання двох різних інструментальних середовищ – CODESYS та OpenPLC – має високу дидактичну цінність, оскільки дозволяє студентам абстрагуватися від інтерфейсу конкретного програмного продукту та зосередитися на універсальних принципах програмування мовами стандарту IEC 61131-3.

Ключовим результатом є формування у студентів здатності до порівняльного аналізу мов програмування на прикладі одного й того самого технологічного процесу («практичного кейсу»). Це дає змогу майбутньому інженеру свідомо обирати оптимальний інструментарій для вирішення конкретних інженерних задач. Фінальний етап інтеграції розроблених алгоритмів із реальним промисловим обладнанням підтвердив, що такий підхід забезпечує формування цілісних компетенцій: від написання коду в режимі симуляції до апаратного налаштування фізичного контролера. Це закладає фундамент для підготовки фахівців, здатних проєктувати сучасні мехатронні комплекси, де механічна конструкція та логіка керування існують як єдине нерозривне ціле.

Заява про використання генеративного ШІ та технологій на основі ШІ в процесі підготовки статті

Під час роботи над цим матеріалом автор використовував можливості моделі Google Gemini 3 Flash як інтелектуального асистента. ШІ-технології були залучені для стилістичної обробки авторських описів методики практичних занять, структурування технічних розділів статті та допомоги в оформленні візуальних матеріалів (описів до ілюстрацій і таблиць).

Після генерації та редагування окремих текстових фрагментів за допомогою ШІ, автор особисто переглянув, скоригував та верифікував увесь зміст на відповідність стандартам і власній викладацькій методиці. Автор несе повну відповідальність за кінцевий зміст публікації та технічну достовірність наведених результатів.

Список використаних джерел

1. Autonomy. (n.d.). *Industrial automation platform*. https://autonomylogic.com/.

2. CODESYS Store International. (n.d.). *CODESYS development system V3*. https://store.codesys.com/en/codesys.html.

3. Zhang, Y., & Wang, J. (2007). *Control system for engine cooling* (U.S. Patent No. 7,270,090). United States Patent and Trademark Office. https://www.freepatentsonline.com/7270090.pdf.

4. Friedrich, H., & Kallenbach, E. (2001). *Electromagnetic actuator for actuating a gas-exchanging valve* (U.S. Patent No. 6,274,954). United States Patent and Trademark Office. https://www.freepatentsonline.com/6274954.pdf.

5. CODESYS Store International. (n.d.). *License agreement for the usage of a CODESYS software or CODESYS software package*. https://store.codesys.com/media/n98_media_assets/files/1101000000/4/CODESYS_EULA_Engineering_en.pdf.

6. Hedrick, J. K., & Cebon, D. (1988). *Method and arrangement for propulsion regulation of an automobile* (U.S. Patent No. 4,765,430). United States Patent and Trademark Office. https://www.freepatentsonline.com/4765430.pdf.

7. Instrumentation Tools. (n.d.). *Multiple pumps control using PLC*. https://instrumentationtools.com/multiple-pumps-control-using-plc/.

8. GCAN PLC & Coupler. (n.d.). *PLC main control GCAN-PLC-301/302*. https://gcanbus.com/products/plc-main-control/.

9. Instrumentation Tools. (n.d.). *PLC program for automatic bottle rejection system*. https://instrumentationtools.com/plc-program-for-automatic-bottle-rejection-system/.

10. Instrumentation Tools. (n.d.). *PLC program for automatic liquid mixing application*. https://instrumentationtools.com/plc-program-for-automatic-liquid-mixing-application/.

11. Instrumentation Tools. (n.d.). *PLC tutorials*. https://instrumentationtools.com/category/plc/.

12. Miyamoto, T., & Sato, H. (2001). *Valve timing control system for internal combustion engine* (U.S. Patent No. 6,325,029). United States Patent and Trademark Office. https://www.freepatentsonline.com/6325029.pdf.

References

1. Autonomy. (n.d.). *Industrial automation platform*. https://autonomylogic.com/.

2. CODESYS Store International. (n.d.). *CODESYS development system V3*. https://store.codesys.com/en/codesys.html.

3. Zhang, Y., & Wang, J. (2007). *Control system for engine cooling* (U.S. Patent No. 7,270,090). United States Patent and Trademark Office. https://www.freepatentsonline.com/7270090.pdf.

4. Friedrich, H., & Kallenbach, E. (2001). *Electromagnetic actuator for actuating a gas-exchanging valve* (U.S. Patent No. 6,274,954). United States Patent and Trademark Office. https://www.freepatentsonline.com/6274954.pdf.

5. CODESYS Store International. (n.d.). *License agreement for the usage of a CODESYS software or CODESYS software package*. https://store.codesys.com/media/n98_media_assets/files/1101000000/4/CODESYS_EULA_Engineering_en.pdf.

6. Hedrick, J. K., & Cebon, D. (1988). *Method and arrangement for propulsion regulation of an automobile* (U.S. Patent No. 4,765,430). United States Patent and Trademark Office. https://www.freepatentsonline.com/4765430.pdf.

7. Instrumentation Tools. (n.d.). *Multiple pumps control using PLC*. https://instrumentationtools.com/multiple-pumps-control-using-plc/.

8. GCAN PLC & Coupler. (n.d.). *PLC main control GCAN-PLC-301/302*. https://gcanbus.com/products/plc-main-control/.

9. Instrumentation Tools. (n.d.). *PLC program for automatic bottle rejection system*. https://instrumentationtools.com/plc-program-for-automatic-bottle-rejection-system/.

10. Instrumentation Tools. (n.d.). *PLC program for automatic liquid mixing application*. https://instrumentationtools.com/plc-program-for-automatic-liquid-mixing-application/.

11. Instrumentation Tools. (n.d.). *PLC tutorials*. https://instrumentationtools.com/category/plc/.

12. Miyamoto, T., & Sato, H. (2001). *Valve timing control system for internal combustion engine* (U.S. Patent No. 6,325,029). United States Patent and Trademark Office. https://www.freepatentsonline.com/6325029.pdf.

Дата першого надходження статті до видання: 01.03.2026
Дата прийняття статті до друку після рецензування: 21.03.2026

TECHNICAL SCIENCES AND TECHNOLOGIES

UDC 378.147:621.3:681.51

Oleksii Samoilenko

PhD in Engineering Sciences, Associate Professor, Associate Professor at the Dept. of Machine Design
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» (Kyiv, Ukraine)
E-mail: o.samoilenko@kpi.ua. **ORCID:** https://orcid.org/0000-0002-0403-1707

THE PROBLEM OF TRAINING FUTURE DESIGN ENGINEERS IN PROGRAMMING CONTROL SYSTEMS FOR TECHNOLOGICAL EQUIPMENT

The article describes a practical approach to training mechanical engineers in programming control systems. The current trend in industry is to replace mechanical, hydraulic, and pneumatic components with electrotechnical systems and software algorithms. For a design engineer, this means a shift from creating purely mechanical structures to developing integrated mechatronic units. This process is seen, for example, in the evolution of automotive systems where physical links are replaced by electric drives and controllers. A similar shift occurs even in switching technology, where electromechanical relays are replaced by solid-state devices to eliminate mechanical wear and increase reliability. The research proposes a training system consisting of nine practical sessions organized in a "1-3-1» structure: introductory tasks, a core block of practical tasks, and a final hardware integration stage. The main feature of this method is the use of a single «practical Case» that students implement using different programming languages. This allows for a direct comparison of different tools for the same technological task. The training is conducted simultaneously in two software environments: OpenPLC and CODESYS. This helps students understand the general principles of the IEC 61131-3 standard rather than just the interface of a specific program. The curriculum covers the Ladder Diagram (LD), Function Block Diagram (FBD), and Structured Text (ST) languages. In the seventh and eighth sessions, students focus on the ST language to implement more complex logic and mathematical operations. The results show that this 9-step method provides students with the skills to perform a full programming cycle, including code, writing in simulation mode, network configuration, and mapping of physical I/O channels of a programmable logic controller. The proposed methodology helps future design engineers choose the most effective programming tools and ensures that the mechanical part of a machine and its control logic are developed as a single system.

Keywords: *programming of technological equipment; electrotechnical systems; programmable logic controller; IEC 61131-3 standard languages; development environment; CODESYS; OpenPLC; emulator; training of design engineers.Fig.: 12. Table: 1. References: 13.*

Самойленко О. В. Проблема підготовки майбутніх конструкторів-машинобудівників до програмування систем керування технологічним обладнанням. *Технічні науки та технології,* 2026. № 2(44). С. 100-115. DOI: https://doi.org/10.25140/2411-5363-2026-2(44)-100-115.