Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ТИПОВИХ ЕЛЕМЕНТІВ ІНТЕГРОВАНИХ IOT СИСТЕМ
RESEARCH AND SOFTWARE IMPLEMENTATION OF TYPICAL ELEMENTS OF INTEGRATED IOT SYSTEMS

Виконав: студент 2 курсу, групи КНТ-214м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Системи штучного інтелекту

ПОЛЯКОВ О.М.
(ПРІЗВИЩЕ та ініціали)
Керівник ПАРХОМЕНКО А.В.
(ПРІЗВИЩЕ та ініціали)
Рецензент ГОЛУБ Т.В.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОЛЯКОВА Олексія Михайловича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація типових елементів інтегрованих IoT систем. Research and Software Implementation of Typical Elements of Integrated IoT Systems

керівник проєкту (роботи) к.т.н., доцент, Пархоменко Анжеліка Володимирівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від "30 вересня 2025 року № 447
2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми проєктування складних IoT систем та постановка завдань дослідження. 2. Матеріали і методи. 3. Методика прототипування та програмна реалізація TEIC. 4. Напрямки використання TEIC.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 01 » жовтня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	3тиждень	Розділ 3
5	Розробка програми.	5-6 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	7-8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка) _____ Олексій ПОЛЯКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Анжеліка ПАРХОМЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 91 с., 8 табл., 53 рис., 6 дод., 32 джерел.

IoT СИСТЕМИ, ІНТЕГРОВАНІ СИСТЕМИ, ПРОГРАМНА РЕАЛІЗАЦІЯ СКІНЧЕНИХ АВТОМАТІВ, МОДЕЛІ СКІНЧЕНИХ АВТОМАТІВ, ПРОГРАМУВАННЯ МОВАМИ СТАНДАРТУ IEC 61131-3.

Об’єкт дослідження – процес програмування складних IoT систем.

Предмет дослідження – методи програмної реалізації типових елементів програмного забезпечення інтегрованих IoT систем.

Мета роботи – зменшення трудомісткості розробки програмного забезпечення інтегрованих IoT систем через розробку шаблонів програмної реалізації типових елементів таких систем на базі теоретико-множинних моделей ієрархічної системи.

Матеріали, методи та технічні засоби: теоретико-множинне моделювання; структурне та об’єктно-орієнтоване програмування; середовище програмування OpenPLC Editor, засіб емуляції OpenPLC Runtime; мови програмування C++, LD, SFC, FBD, ST, IL; персональний комп’ютер з процесором AMD Ryzen 7 7730U під управлінням операційної системи Microsoft Windows 11.

Результати. Проведено моделювання та аналіз функціональної структури складної IoT системи, визначені типові елементи. Для цих елементів розроблені та досліджені програмні шаблони, розроблено методику їх застосування, наведено приклад використання.

Висновки. Розроблені програмні шаблони типових елементів інтегрованих IoT систем дозволяють зменшити трудомісткість розробки програмного забезпечення таких систем.

Галузь використання – процеси програмування інтегрованих IoT систем.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 91 pages, 8 tables, 53 figures, 6 appendixes, 32 sources.

IoT SYSTEMS, INTEGRATED SYSTEMS, SOFTWARE IMPLEMENTATION OF FINITE AUTOMATION MACHINES, FINITE AUTOMATION MODELS, PROGRAMMING IN LANGUAGES OF THE IEC 61131-3 STANDARD.

The object of the study is the process of programming complex IoT systems.

The subject of the study is methods of software implementation of typical elements of integrated IoT systems software.

The purpose of the work is to reduce the labor intensity of developing integrated IoT software through the development of templates for software implementation of typical elements of such systems based on set-theoretic models of a hierarchical system.

Materials, methods and technical means: set-theoretic modeling: structural and object-oriented programming, OpenPLC Editor programming environment, OpenPLC Runtime emulation tool; programming languages C++, LD, SFC, FBD, ST, IL; personal computer with an AMD Ryzen 7 7730U processor running the Microsoft Windows 11 operating system.

Results. The functional structure of a complex IoT system was modeled and analyzed, typical elements were identified. Software templates were developed and researched for these elements, a methodology for their application was developed, and an example of use was given.

Conclusions. The developed software templates of typical elements of integrated IoT systems allow reducing the labor intensity of software development of such systems by 10-15 percents.

Field of application – programming processes of integrated IoT systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

DCS – Distributed Control System;

IDE – Integrated Development Environment;

IEC – International Electrotechnical Commision;

IL – Instruction List;

FBD – Function Block Diagram;

HMI – Human Machine Interface;

LD – Ladder Diagram;

MMI – Man Machine Interface;

POU – Program Organization Unit;

PLC – Programing Logic Controller;

SCADA – Supervisory Control and Data Acquisition;

SFC – Sequential Function Chart;

SoC – System-on-Chip;

ST – Structured Text;

UDB – Universal Digital Block;

АЦП – аналого-цифровий перетворювач;

ВС – вбудована система;

ІКС – інформаційно-керуючи системи;

ККС – когнітивні-керуючі системи;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ПСнК – програмована система на кристалі;

ТЕІС – типовий елемент інтегрованої системи;

ЦАП – цифро-аналоговий перетворювач.

ВСТУП

Інтернет речей (IP, англ. Internet of Things, IoT) — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку. Окрім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі [1].

Промисловий інтернет речей (англ. industrial internet of things, IIoT) — система об'єднаних комп'ютерних мереж та підключених до них промислових (виробничих) об'єктів із вбудованими датчиками та програмним забезпеченням для збору та обміну даними, з можливістю віддаленого контролю та управління в автоматизованому режимі. Таке підключення дозволяє збирати, обмінюватися та аналізувати дані, що потенційно сприяє вдосконаленню та оптимізації керування процесами, підвищенню продуктивності та ефективності виробництва, а також іншим економічним перевагам. IIoT — це результат еволюції розподіленої системи керування (DCS) [2].

Чисельні публікації по тематиці IoT та IIoT присвячені переважно питанням забезпечення мережевого доступу [3] до розподілених давачів та виконавчих механізмів, особливостей IIoT [4], програмування пристроїв IoT [5]. В той самий час використання отриманих даних відбувається на інформаційному рівні в рамках інформаційно-керуючих систем. Системи що використовують вищі форми знань не мають шаблонів для програмної реалізації типових елементів що приводить до збільшення трудомісткості розробки програмного забезпечення.

Одним із класів систем управління є інтегровані системи [6]. Як мінімум дві підсистеми у такій системі мають загальний елемент, який виконує різні функції у цих системах [7]. Наприклад, в одній системі – функцію керу-

ючого пристрою, а в другій – об'єкт управління. Така взаємодія підсистем дозволяє створювати системи з ієрархією управлінь, за допомогою якої виконується адаптація структури та параметрів системи до зміни цілей їх функціонування та параметрів зовнішніх впливів на систему [8].

Рівні інтегрованої системи відрізняються формою знань що обробляються, логікою дій системи зумовленою результатами цієї обробки. Алгоритми обробки та синтезу знань реалізовані програмно та виконуються у середовищі вузлів системи, що мають обчислювальний ресурс – комп’ютерів, контролерів та PLC.

Тому, тема роботи є актуальною.

Мета роботи – зменшення трудомісткості розробки програмного забезпечення інтегрованих IoT систем через розробку шаблонів програмної реалізації типових елементів таких систем на базі теоретико-множинних моделей ієрархічної системи.

1 АНАЛІЗ ПРОБЛЕМИ ПРОЄКТУВАННЯ СКЛАДНИХ ІОТ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз структури IoT системи

Узагальнена структура апаратного забезпечення IoT системи наведена на рис. 1.1.

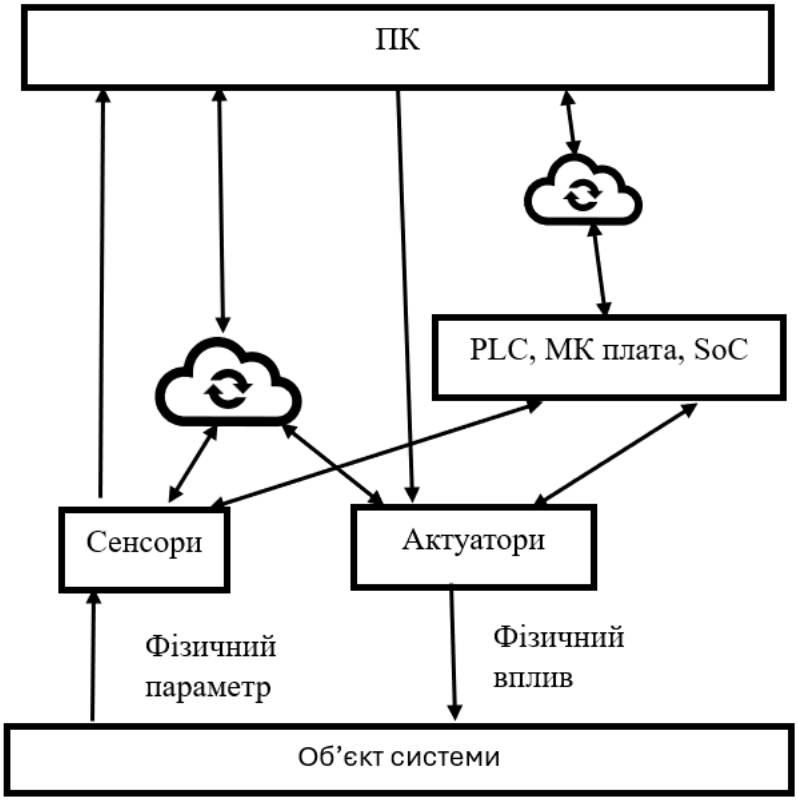


Рисунок 1.1 – Узагальнена структурна схема IoT системи

Основні елементи IoT системи це сенсори, актуатори, пристрої з обчислюваним ресурсом (PLC, МК плати, SoC та ПК) та канали інформаційно-керуючих мереж. Сенсори перетворюють фізичний параметр на сигнал який надходить до пристрою керування. У деяких випадках сенсор додатково переробляє сигнал у данні, які передаються через інформаційно-керуючи мережі у вигляді повідомлень до ПК.

Як правило, на входи актуатору надходить сигнал від керуючого пристрою, який визначає рівень фізичного впливу на об’єкт системи. У деяких

випадках на входи актуатору сигнал надходить у формі даних які передаються по інформаційно-керуючий мережі. Тобто, у даному випадку, трансформація даних в параметри впливу відбувається у актуаторі.

Відповідно з технічним завданням на кваліфікаційну роботу (Додаток А) розглядається IoT система у якої пристроєм з обчислювальним ресурсом є PLC.

У архітектурі IoT на фізичному рівні та рівні додатків важливу роль відіграють моделі сумісності IoT, платформи, технології та протоколи передачі даних [9]. До ключових понять IoT відносять також «Аналітичний фактор» який представляє програмні системи, які аналізують дані, отримані від IoT-пристроїв. Аналітика використовується у великій кількості сценаріїв – наприклад, для прогнозування технічного обслуговування [9]. На цьому рівні у поточний час відсутні узагальнюючі моделі інтелектуальної/ «розумної»/когнітивної обробки знань та формування відповідних керуючих впливів на об’єкт системи. Тому алгоритми обробки є індивідуальними для кожного завдання користувача, що пов’язане зі значними трудовитратами у процесі проєктування IoT системи.

1.2 Моделі знань та керуючих структур

У багатьох випадках удосконалення алгоритмів використання даних в IoT системах передбачає, що використання додаткових знань про об'єкт та пристрій керування покращує якість керування. Ці знання існують у різних формах. Ієрархічний зв'язок між цими формами візуалізовано на рисунку 1.2 пірамідою форм знань DIKW або DIKUW, де D – дані, I – інформація, K – знання, U – розуміння, W – мудрість [10], [11].

Використання знань з метою управління системним об'єктом також передбачає ієрархію інструментів управління, які враховують характеристики об'єкта управління та реалізують різні методи та моделі управління [8]. Доте-

пер для управління найбільш широко використовувалися знання у вигляді даних та інформації. Відповідні системи називаються інформаційно-керуючими системами (ІКС). Системи управління, що використовують вищі форми знань (знання, розуміння та мудрість) та мають когнітивні здібності сприйняття, навчання, планування та міркування, називатимуться когнітивними [12]. «Cognitio» в перекладі з латини означає «знання» [13]. Когнітивна наука є частиною авторитетного вектору технологічного розвитку світової економіки NBIC (Nano - Bio - Info - Cogni) [14].

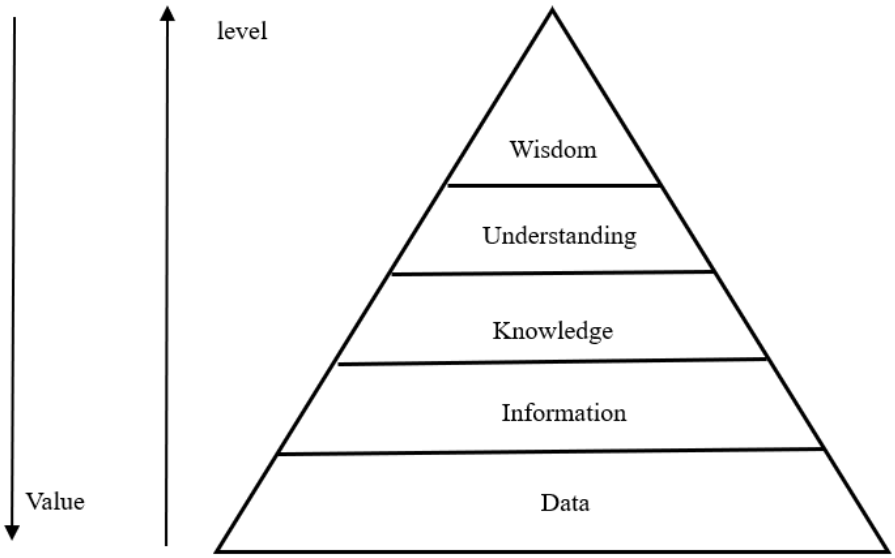


Рисунок 1.2 – Піраміда форм знань

У публікаціях про когнітивні системи немає єдності не лише щодо питання, що таке така система, але й що слід вважати знаннями, розумінням та мудрістю технічних систем, яким набором когнітивних здібностей повинна володіти система, щоб вважатися когнітивною.

Форми, одиниці та моделі знань, що вже використовуються в IoT системах наведені в таблицях 1.1 та 1.2 [10]. Нижчі форми знань добре вивчені та використовуються в ІКС. Однак, оскільки ІКС включені як підсистеми до ККС, їх вивчення також є необхідним. Обробка цих форм знань може здійснюватися без використання когнітивних здібностей.

Таблиця 1.1 – Характеристики сигналів та даних в Інтернеті речей

Форми знань	Одиниці знань	Операції зі знаннями / когнітивні здібності	Модель знань
Сигнал – це форма відображення значення параметра деякого фізичного процесу.	Значення фізичного параметра, що несе інформацію в певний момент часу.	Коефіцієнт підсилення, фільтрація /немає.	Тип інформаційного параметра; коефіцієнт підсилення; параметри фільтра; діапазон допустимих значень.
Дані – це результат реєстрації та первинного перетворення сигналу в системі.	Двійковий код сигналу, системна мітка часу в момент виявлення та номер вхідного каналу блоку керування, що зберігаються на машинному носії.	Аналогова - цифрове перетворення сигналу; призначення адреси системної пам'яті; цифрова фільтрація /немає.	Розрядність, опорна напруга/струм, інтервал дискретизації АЦП; метод адресації даних; параметри цифрового фільтра.

Таблиця 1.2 – Форми інформації

Форми знань	Одиниці знань	Операції зі знаннями/когнітивні здібності	Модель знань
1	2	3	4
Інформація – це результат інтерпретації даних у системі або обробки іншої інформації.	Первинний тег: назва, місце, астрономічний час, похибка вимірювання та значення фізичної величини, представленої даними.	Перетворення кодів даних, часу детектування, номера вхідного каналу сигналу /немає	Таблиця відповідності: канали даних типу фізичної величини, місцю вимірювання, одиницям вимірювання. Таблиця відповідності системних та астрономічних параметрів часу

Кінець табл. 1.2

1	2	3	4
	Вторинний тег: назва та значення характеристики масиву, посилання на назву та параметри масиву.	Обробка масиву інформації: знаходження характеристики масиву (наприклад, середнє арифметичне, максимум, мінімум) /немає	Початковий масив, алгоритм отримання характеристик
	Первинний масив: назва масиву та параметри.	Створення та/або виділення масиву /немає	Початкова адреса в пам'яті, розмірність та обсяг масиву, алгоритм обробки
	Вторинний масив: назва масиву та параметри.	Обробка масиву інформації методом апроксимації, інтерполяції, екстраполяції, спектрального аналізу. Знаходження коефіцієнтів кореляції фізичних величин /немає.	Вихідний масив, параметри обробки, параметри вторинного масиву.

Приклади одиниць знань наведено в табл. 1.3. Приклади є умовними та демонструють структуру одиниць знань.

Піраміда знань використовується у IoT системі для аналізу поточної ситуації та формування відповідної діяльності направленої на досягнення системою певної мети її функціонування. При цьому програмні засоби що формують діяльність системи також мають ієрархічну структуру. Окремі рівні цієї структури реалізують автоматну поведінку.

Теорія автоматів є загальновизнаною основою інформатики. Потреба в моделі автомата виникла через зростання складності завдань керування технічними системами. До появи автомата для такого керування використовувалися лише поточні значення сигналів від об'єкта керування.

Таблиця 1.3– Приклади знань

Форма знання	Одиниця знань
1	2
Сигнал	Напруга сигналу: +1,24 мВ
Дані	Двійковий код сигналу: 100111001011; системний код часу на момент вимірювання: 1110101000011100; номер вхідного каналу блоку керування: 1000111; початкова адреса даних у пам'яті блоку керування: FFF7A123hex
Інформація	Основний тег: Назва – температура; місце – ізоляція верхнього шару обмотки трансформатора, астрономічний час – 12-30; похибка виміру – не більше 10% та значення фізичної величини – 92 С
	Вторинний тег: назва – середня температура ізоляції верхнього шару обмотки трансформатора за останню годину; значення – 87 С; характеристики масиву: посилання на назву – «температура ізоляції верхнього шару обмотки трансформатора з 0:00 до 01:00 1 серпня; параметри масиву: кількість елементів: 60.
	Основний масив: назва: «температура (виміряна) обмотки за останню годину»; параметри масиву: кількість елементів -10.
	Вторинний масив: назва: «температура (інтерпольована) обмотки за останню годину»; та параметри масиву: кількість елементів -60.

Модель автомата дозволила враховувати попередні значення сигналів для цілей керування, і таким чином розширилася інформаційна база для прийняття керуючих рішень.

Теоретико-множинна модель автомата, як перетворювача, має вигляд кортежу (1.1) [16], [17]:

$$< S, X, Y, s0, \delta, \lambda >, \tag{1.1}$$

де S – скінченна непорожня множина станів;

X – скінченна непорожня множина входів (вхідний алфавіт);

Y – скінченна непорожня множина виходів (вихідний алфавіт);

$s0$ – початковий стан;

δ – функція переходу;

λ – вихідна функція.

Функція переходу має вигляд відображення (1.2):

$$\delta: S(t) \times X \to S(t+1). \tag{1.2}$$

Функція виходу для автомата Мура [17] має вигляд відображення (1.3):

$$\lambda: S \to Y, \tag{1.3}$$

а для автомата Мілі [18] вона має вигляд відображення (1.4):

$$\lambda: S \times X \to Y. \tag{1.4}$$

Таким чином, вихід автомата Мура не залежить від входу, яким активується заданий стан. У той час як у автоматі Мілі кожен вхід стану може відповідати власному виходу.

Автомати, представлені виразами (1.1) – (1.4), називатимемо класичними. Такі машини, як буде показано нижче, мають ряд обмежень і недостатньо ефективні для моделювання ієрархічних інтегрованих систем керування.

Тому в наступних розділах запропоновано ряд рішень, які долають обмеження класичних автоматів, мають розширену функціональність та підвищують ефективність моделювання таких систем.

1.3 Аналіз мов програмування

В процесі розробки програмного забезпечення IoT систем використовуються як мови загального призначення Python, Java, JS, C/C++, а також спеціалізовані мови програмування PLC за стандартом IEC 61131-3 [19]. З урахуванням вимог технічного завдання, більш детально розглянемо мови за цим стандартом. Вони повинні корелювати з предметною областю «мало програмуючого професіонала», який до того ж має звичку працювати зі схемами.

Мови програмування, визначені стандартом IEC 61131–3 наведені в табл. 1.4 [19] - [22].

Таблиця 1.4 – Характеристика мов програмування за стандартом IEC 61131–3

Позначення	Найменування	Тип мови	Область ефективного застосування
LD	Ladder Diagram, драбинні діаграми	Графічна	Логіка взаємного блокування
SFC	Sequential Function Chart, послідовні функціональні кроки	Графічна	Функціональна специфікація системи, керування послідовністю кроків
FBD	Function Block Diagram, функціональні блокові діаграми	Графічна	Процеси безперервного керування, складні обчислювання
ST	Structured Text, структурований текст	Текстова	Циклічні та розгалужені алгоритми, робота зі складними типами даних
IL	Instruction List, список інструкцій	Текстова	Там де потрібна мінімізація часу виконання та обсягу памяті

Мова LD – представляє логіку програми у формі, схожій на електричну схему з ланцюгами, що складаються з контактів і обмоток (котушок) реле.. Програма на мові LD складається з рангів. Кожен ранг – це програмний аналог логіки одного електричного ланцюга [23].

Ранг містить умовні і виконавчі інструкції і еквівалентний операторові умовного виконання IF умова THEN дія, де умова представлена однією або декількома умовними інструкціями, логічні зв'язки між якими задані графічно гілками рангу, а дія представлена однією або декількома виконавчими інструкціями, послідовність виконання яких також задана графічно гілками рангу. Приклад програми мовою LD, що виконує функції лічильника приведено на рис. 1.3.

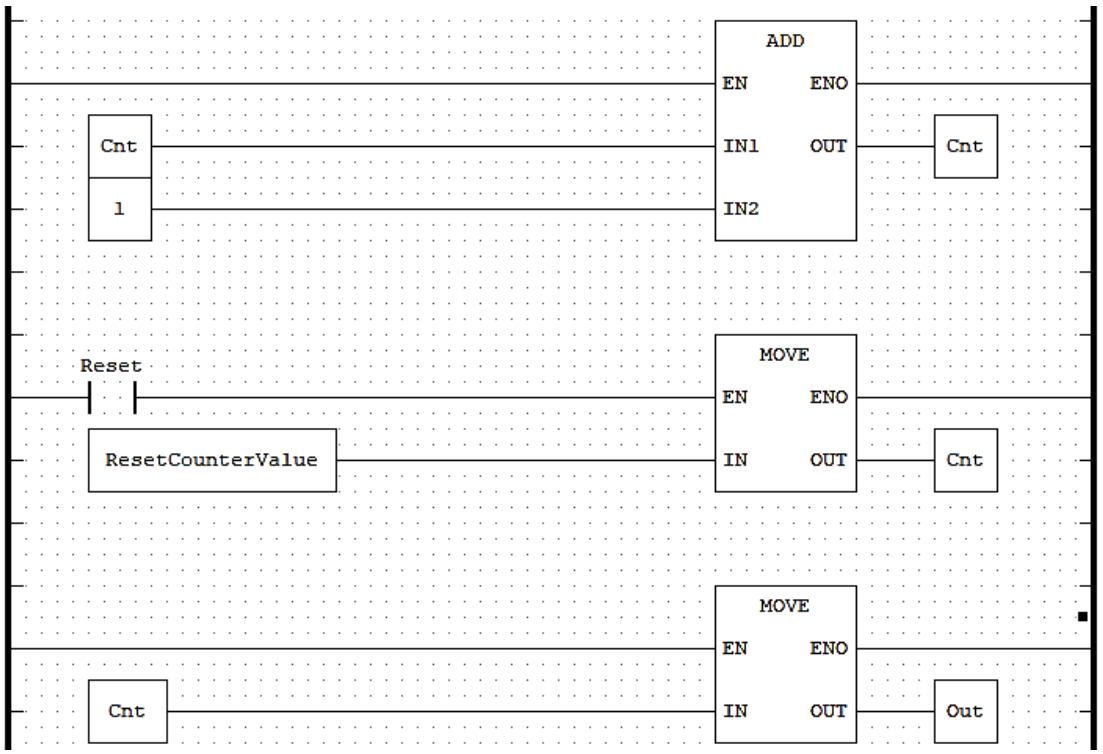


Рисунок 1.3 – Лічильник мовою LD

Програма мовою FBD візуально нагадує електричну схему цифрового вузла Вона може містити декілька підсхем які називають ланцюгами. Програми, відповідні ланцюгам виконуються послідовно зверху-вниз в порядку

розташування в діаграмі FBD. Приклад програми мовою FBD, що виконує функції лічильника приведено на рис. 1.4.

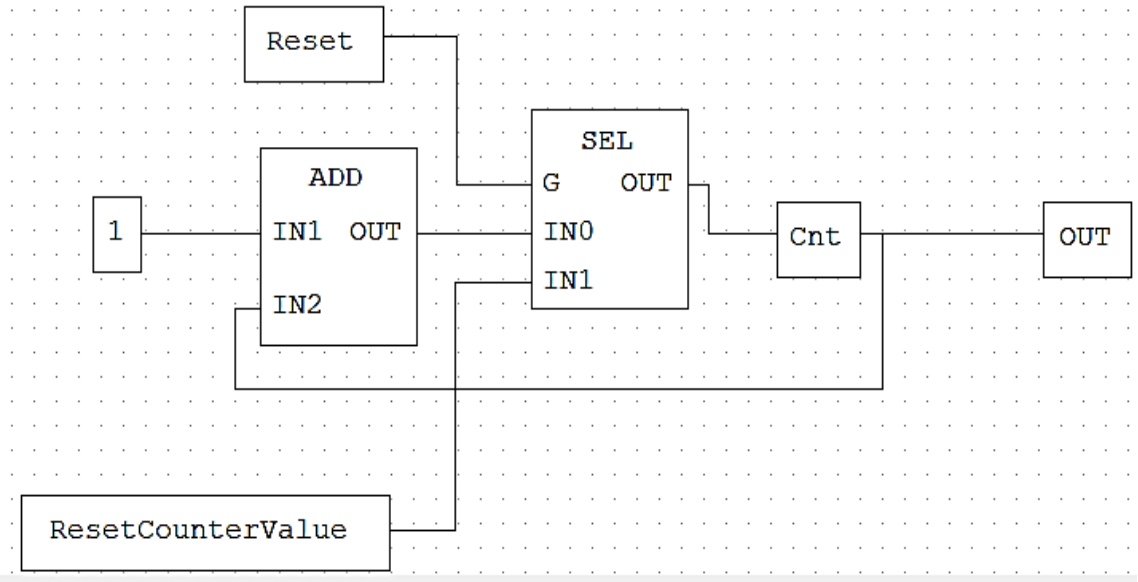


Рисунок 1.4 – Лічильник мовою FBD

Мова SFC представляє логіку програми у формі направленого графа. Вершини графа відповідають крокам, а дуги – переходам процесу керування. Процес виконання програми SFC можна представити як процес переходу від кроку до кроку. Коли крок стає активним, він ініціює виконання певних дій або може викликати дочірню SFC програму. Інтервал часу, в якому виконуються дії, може не співпадати з інтервалом активності кроку і задається кваліфікаторами кроку. Переходи SFC програми описують умови припинення активності поточного кроку і передачі її наступному кроку. Дія кроків і логічні умови переходу програмуються на мовах LD, FBD, ST, IL [23].

Приклад програми мовою SFC, що виконує функції лічильника приведено на рис. 1.5.

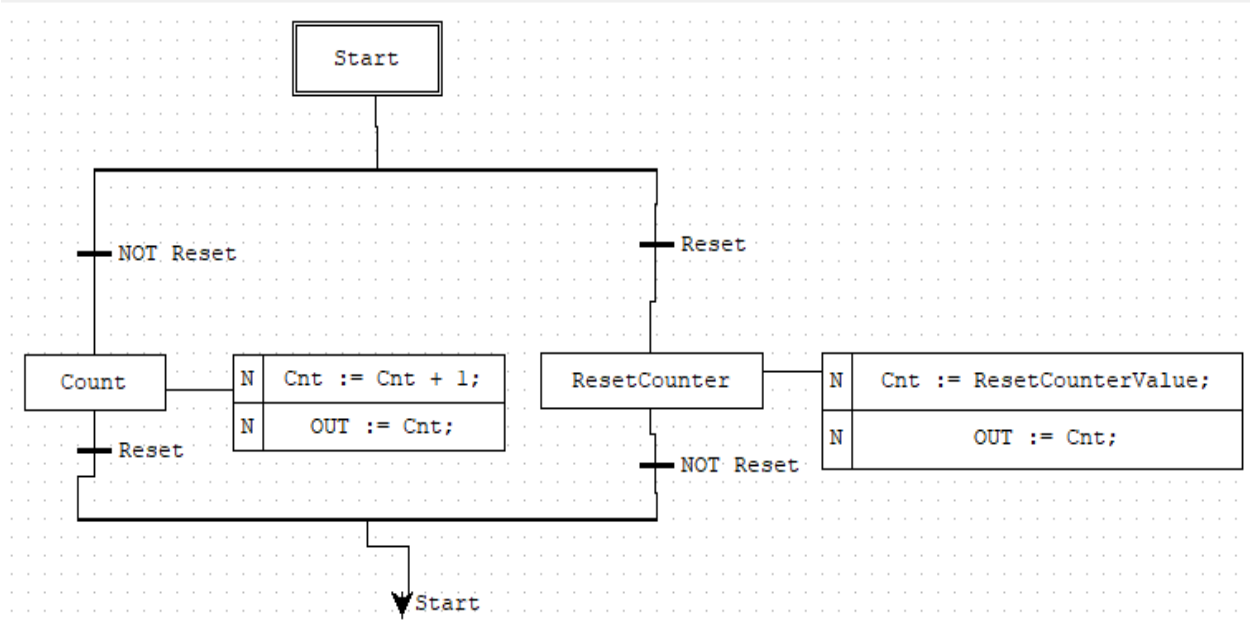


Рисунок 1.5 – Лічильник мовою SFC

Текстова мова ST за синтаксисом на набором операторів нагадує мову Паскаль. Програма ST – це список операторів, що закінчуються крапкою з комою. Мова ST використовується для опису умов переходів і дій усередині кроків програми SFC.

Приклад програми мовою ST, що виконує функції лічильника приведено на рис. 1.6.

```
1  IF Reset THEN
2      Cnt := ResetCounterValue;
3  ELSE
4      Cnt := Cnt + 1;
5  END_IF;
6
7  Out := Cnt;
```

Рисунок 1.6 – Лічильник мовою ST

Мова IL – це «асемблероподібна» мова. Програма на IL є послідовністю інструкцій. Інструкція IL може містити від одного до чотирьох полів: метка, мнемокод операції, операнди та коментар.

Приклад програми мовою IL, що виконує функції лічильника приведено на рис. 1.7.

```
LD Reset
JMPC ResetCnt

(* increment counter *)
LD Cnt
ADD 1
JMP QuitFb

ResetCnt:
(* reset counter *)
LD ResetCounterValue

QuitFb:
(* save results *)
ST Cnt
ST Out

```

Рисунок 1.7 – Лічильник мовою IL

Слід відзначити, що у додатку керування можуть використовуватися одночасно декілька мов програмування.

Розробка стандарту IEC 61131–3 дозволило уніфікувати мови програмування за рахунок визначення загальних вимоги до мов програмування та видів мов для програмованих контролерів. Стандарт IEC 61131–3 також породив новий, різновид програмних продуктів – універсальний засіб програмування на стандартних мовах за стандартом IEC 61131–3, наприклад пакети Codesys фірми 3s – Smart Software Solution [24], OpenPLC [25], ISaGRAF [26].

1.4 Постановка завдань дослідження

Сучасні IoT системи це окремий вид розподілених систем які виконують моніторинг стану та параметрів об’єкту системи та діяльність що направлена на зміну стану об’єкту системи у бажаному напрямку.

Питання обміну даними вузла IoT системи що має обчислюваний ресурс із периферійними пристроями вирішується типовим шляхом з використанням стандартних протоколів обміну.

У той самий час, обробка отриманих даних у кожному випадку виконується по індивідуальним алгоритмам, що підвищує трудомісткість проєктування системи.

В результаті аналізу визначено що процес обробки отриманих даних спирається піраміду форм знань DIKUW а діяльність системи виконується в рамках інтегрованої системи. Кожний рівень такої системи реалізує певну поведінку та пов’язаний із суміжними рівнями.

Завдання дослідження полягає в тому щоби знайти типовий елемент такої системи використовуючи моделі скінчених автоматів. З урахуванням вимоги технічного завдання про реалізацію системи на базі PLC програмна реалізація типового елемента повинна бути розроблена мовами програмування та елементами програмної організації за стандартом IEC 61131-3.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Програмна модель керуючого автомату мовою С

Вхідними даними для побудови програмної моделі скінченого автомата є його граф або таблиці переходів та виходів, початковий стан автомата. Елементам множин теоретико-множинного кортежу автомата відповідають змінні процедури автомату:

- S_i ($i = 0, I$) – описують множину станів; $S_i = 0$, якщо стан пасивний; $S_i = 1$, якщо стан активний, де I – кількість станів автомата;
- X_j ($j = 1, J$) – описують множину подій (входів); $X_j = 0$, якщо подія відсутня; $X_j = 1$, якщо подія відбувається, де J – кількість подій автомата;
- Y_k ($k = 1,K$) – описують множину дій (виходів); $Y_k = 0$ якщо дія Y_k не виконується; $Y_k = 1$, якщо дія виконується.

Тобто усі ці змінні відносяться до бінарного типу даних.

Програмна модель скінченого автомата містить блоки задання початкового стану, виконання дій (виходів) у поточному стані, виконання переходу у новий стан. Процедура автомата виконується циклічно.

Для задання початкового стану автомату S_i виконують $S_i = 1$, а для усіх інших змінних що описують множину станів $S_r = 0$, ($r \neq i$). Блок задання початкового стану виконується одноразово.

Блок виконання дій у поточному стані для автомата Мура містить I операторів умовного виконання: if (Si == 1) Yi = 1 Else Yi = 0.

Блок виконання переходу у новий стан для автомата Мура містить M операторів умовного виконання, де M – кількість дуг у графі автомата. Так для переходу з i-го стану в p-й під дією входу X_j оператор програми має вигляд:

if (Si==1)&&(Xj==1) (Sp = 1; Si = 0).

На рис. 2.1 наведено приклад задання керуючого автомата за допомогою графу.

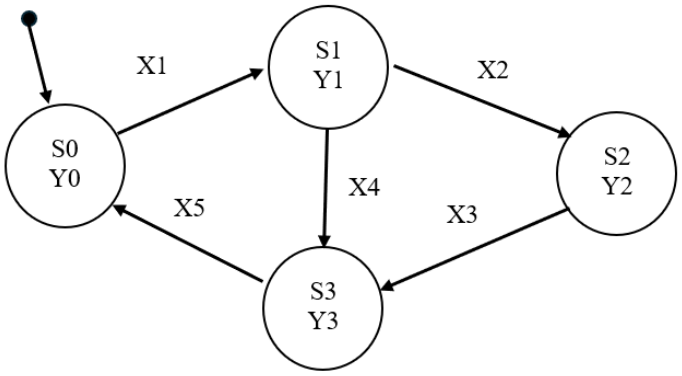


Рисунок 2.1 – Приклад графу керуючого автомату

На рис.2.2 той самий керуючий автомат задано у табличній формі.

Поточний стан	Новий стан			
	S0	S1	S2	S3
S0		X1		
S1			X2	X4
S2				X3
S3	X1			

а

Стан	Вихід
S0	Y0
S1	Y1
S2	Y2
S3	Y3

б

а – таблиця переходів; б – таблиця виходів

Рисунок 2.2 – Приклад задання керуючого автомату у формі таблиць

На рис. 2.3 наведено схему під’єднання об’єкту керування до плати мікроконтролеру.

Рисунок 2.3 – Схема під’єднання об’єкту керування до плати мікроконтролеру

На рис.2.4 наведено текст програми що реалізує керуючий автомат з урахуванням даних рис. 2.1 – 2.3.

```
boolean x1, x2, x3, x4, x5;//змінні входов
boolean s0, s1, s2, s3;//змінні станів
const int y0 =10;//визначення пінів виходів
const int y1 =11;//визначення пінів виходів
const int y2 =12;//визначення пінів виходів
const int y3 =13;//визначення пінів виходів
const int pinx1=5;//визначення пінів входів
const int pinx2=6;//визначення пінів входів
const int pinx3=7;//визначення пінів входів
const int pinx4=8;//визначення пінів входів
const int pinx5=9;//визначення пінів входів
void setup() {
  pinMode(pinx1,INPUT);//визначення входів
  pinMode(pinx2,INPUT);//визначення входів
  pinMode(pinx3,INPUT);//визначення входів
  pinMode(pinx4,INPUT);//визначення входів
  pinMode(pinx5,INPUT);//визначення входів
  pinMode(y0,OUTPUT);//визначення виходів
  pinMode(y1,OUTPUT);//визначення виходів
  pinMode(y2,OUTPUT);//визначення виходів
  pinMode(y3,OUTPUT);//визначення виходів
  s0=1;//початковий стан
  s1=0;//початковий стан
  s2=0;//початковий стан
  s3=0;//початковий стан
}

void loop() {
if (s0==1) digitalWrite(y0, HIGH); else digitalWrite(y0, LOW);//форму-
вання виходу у стані s0
```

```
if (s1==1) digitalWrite(y1, HIGH); else digitalWrite(y1, LOW);//форму-
вання виходу у стані s1
if (s2==1) digitalWrite(y2, HIGH); else digitalWrite(y2, LOW);//форму-
вання виходу у стані s2
if (s3==1) digitalWrite(y3, HIGH); else digitalWrite(y3, LOW);//форму-
вання виходу у стані s3
  x1=digitalRead(5);//читання x1
  x2=digitalRead(6);//читання x2
  x3=digitalRead(7);//читання x3
  x4=digitalRead(8);//читання x4
  x5=digitalRead(9);//читання x5
  if((s0==1)&&(x1==1)) s1=1;s0=0;//перехід s0s1
  if((s1==1)&&(x2==1)) s2=1;s1=0;//перехід s1s2
  if((s1==1)&&(x4==1)) s3=1;s1=0;//перехід s1s3
  if((s2==1)&&(x3==1)) s3=1;s2=0;//перехід s2s3
  if((s3==1)&&(x5==1)) s0=1;s3=0;//перехід s3s0
}
```

Рисунок 2.4. – Текст програми керуючого автомату

Програма рис. 2.4 написана мовою С у середовищі Arduino IDE [27]. Відзначимо що є PLC під торговою маркою Controlline які сумісні с Arduino IDE [28].

Структура програми відображає логіку конкретного автомата керування. Тобто для програмної реалізації іншої логіки потрібна інша програма, що підвищує загальну трудомісткість процесу програмування.

Ідея уніфікованої програми керуючого автомату полягає в тому щоб логіку роботи керуючого автомата подавати у вигляді даних які обробляє ця програма. Для налагодження програми на логіку конкретного автомату використовуються наступні дані:

- назва початкового стан;
- таблиця виходів (назва стану/вихід який виконується в цьому стані). Тобто формується «1» по цьому виходу на інших виходах-«0»);

- таблиця відповідності пінів плати мікроконтролера входам автомата;
- таблиця відповідності пінів плати мікроконтролера виходам автомата;
- таблиця переходів автомата – масив, в якому кожен рядок відповідає поточному стану та підмножині входів автомата, по яких відбувається перехід автомата в новий стан.

Текст уніфікованої програми керуючого автомату наведено у додаку Б. Програма написана універсально, але через архітектуру Arduino (AVR, 8-біт) та спосіб зберігання таблиць є точні межі щодо кількості станів, входів та виходів автомата.

Максимальна кількість станів визначається таблицею OUTPUTS (кожен елемент масиву – один стан) яка описується константою const uint8 Тому теоретично максимальна кількість станів дорівнює 255. Практично, через невелику пам'ять системи, кількість станів обмежена до 100.

Обмеження входів (inputs) також обмежено типом константи маски входів і дорівнює 16.

Обмеження з виходів (outputs) визначено розрядністю масива OUTPUT_PINS[] = {10, 11, 12}. Тобто 3. Теоретично індекс виходу має тип uint8, тому можливо використовувати до 256 виходів, але з урахуванням кількості пінів мікроконтролерної плати -20-30 виходів.

Четверте обмеження це обмеження переходів (transitions). У програми можливо до 16 переходів з одного стану. Для збільшення до кількості переходів із станів до 255 потрібно змінити формат переходів використовуючи тип uint32

2.2 Програмна модель керуючого автомату мовами SFC та LD

Як відзначалось у розділі 1, у випадку проєктування IIoT систем використовуються PLC які програмуються тільки мовами стандарту IEC 61131-3.

Тому розглянемо методи програмної реалізації керуючих автоматів графічними мовами програмування. Програмування здійснено у середовищі OpenPLC [25]. Змінні SFC програми керуючого автомату наведено на рис. 2.5.

Візуальне представлення (схема) SFC програми керуючого автомату заданого графом рис. 2.1 наведено на рис. 2.6. Це представлення складається з таблиці змінних, та схеми програми. Схема містить стани(кроки) та переходи з умовами у вигляді входів. У станах виконуються певні дії які визначаються активністю виходів.

Описание: ClasFSM Фильтр класса: Все

#	Имя	Класс	Тип	Адрес	Исходное значение
1	x1	Вход	BOOL		0
2	x2	Вход	BOOL		0
3	x3	Вход	BOOL		0
4	x4	Вход	BOOL		0
5	x5	Вход	BOOL		0
6	start	Вход	BOOL		0
7	y0	Выход	BOOL		0
8	y1	Выход	BOOL		0
9	y2	Выход	BOOL		0
10	y3	Выход	BOOL		0

Рисунок 2.5 – Змінні LD програми керуючого автомату

Іншою поширеною мовою програмування PLC є мова LD. Програма мовою LD складається з рангів. Для реалізації керуючого автомата використовуються три типи рангів:

- установка початкового стану автомата;
- активація дії автомата у поточному стані;
- перехід автомату у новий стан при активності певного входу автомату.

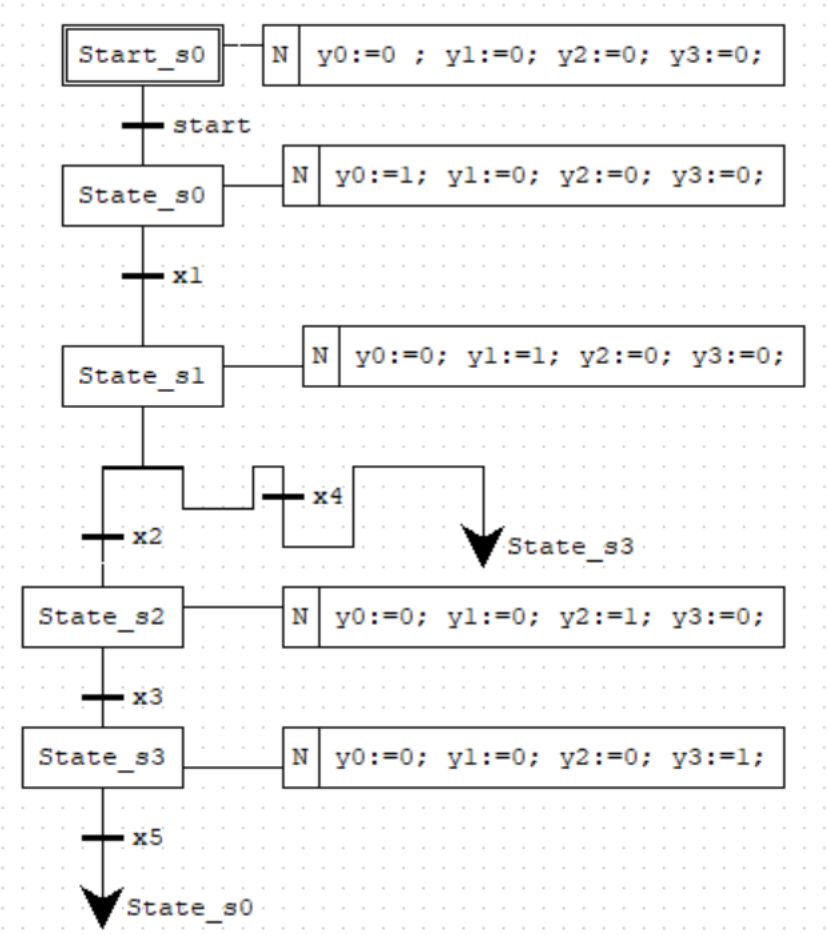


Рисунок 2.6 – Візуальне представлення (схема) SFC програми керуючого автомату

Змінні LD програми керуючого автомату наведено на рис. 2.7. Візуальне представлення (схема) LD програми керуючого автомату заданого графом рис. 2.1 наведено на рис. 2.8.

Перший ранг цієї програми забезпечує установку автомату у початковий стан s0 за рахунок попередньої уставки змінної first_scan. Ця змінна скидається у останньому рангу першого скану програми. Тому установка початкового стану виконується тільки один раз.

Таким чином, програмна реалізація класичні керуючих автоматів можлива як універсальними так і спеціалізованими мовами програмування.

LD_FSM

Описание: Фильтр класса: Все

#	Имя	Класс	Тип	Адрес	Исходное значение
1	first_scan	Локальный	BOOL		1
2	x1	Вход	BOOL		0
3	x2	Вход	BOOL		0
4	x3	Вход	BOOL		0
5	x4	Вход	BOOL		0
6	x5	Вход	BOOL		0
7	s0	Локальный	BOOL		0
8	s1	Локальный	BOOL		0
9	s2	Локальный	BOOL		0
10	s3	Локальный	BOOL		0
11	y0	Выход	BOOL		0
12	y1	Выход	BOOL		0
13	y2	Выход	BOOL		0
14	y3	Выход	BOOL		0

Рисунок 2.7 – Змінні LD програми керуючого автомату

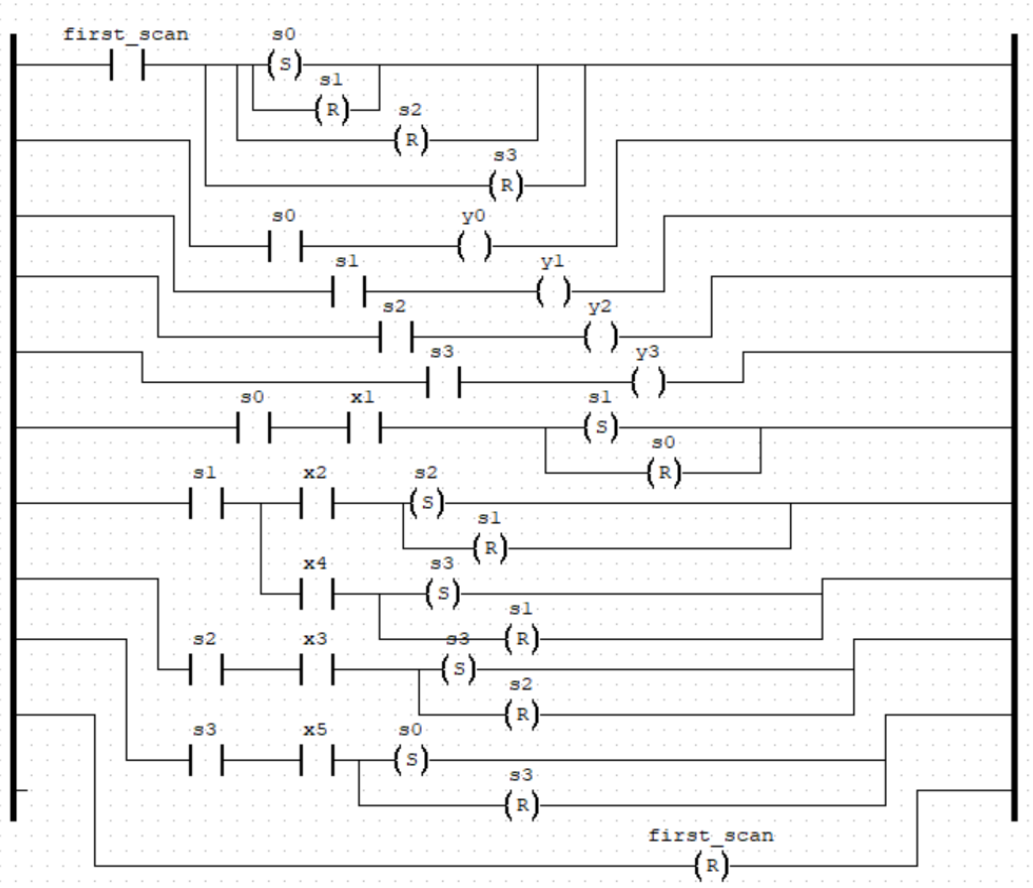


Рисунок 2.8 – Візуальне представлення LD програми керуючого автомату

Для розуміння розширень автоматів, описаних у наступних розділах, слід звернути увагу на такі основні властивості класичних автоматів [15].

Властивість 1. Усі елементи множин X, Y, S є елементами двійкового типу даних. Наприклад, стан s_i ($s_i \in S$) може бути пасивним або активним.

Властивість 2. Під час активності стану вихід, пов'язаний з ним, є активним. Часові інтервали виходу та активності стану збігаються.

Властивість 3. Множини S, X, Y, вихідні функції λ, переходи δ та початковий стан s0 автомата не змінюються під час його роботи.

Властивість 4. У кожен момент роботи автомата активним є один і тільки один стан. У початковий момент роботи автомата активним є стан $s0$.

Властивість 5. Вхід $x_i \in X$ інтерпретується як певна функція параметрів об'єкта керування, середовища та команд оператора. Ця функція не змінюється під час роботи автомата.

Властивість 6. Активність виходу $y_j \in Y$ ініціює процес застосування дії до об'єкта керування. Параметри цієї дії не змінюються під час роботи автомата.

Властивість 7. У будь-який момент часу активним може бути лише один вхід, або активних входів немає.

2.3 Типовий елемент інтегрованої системи

Модель класичного автомату є простою, але використання її для моделювання складних алгоритмів керування системою призводить до збільшення розмірності цієї моделі. Крім того, класична модель не передбачає структурну адаптацію автомата у процесі його функціонування [6].

Для побудови типового елемента інтегрованої системи скористаємось моделлю автомата з додатковим безліччю управлінь та безліччю функцій автомата у його станах [29] та моделями операційних автоматів [23]. Елемент множини управлінь визначає можливості виходу автомата з поточного стану

при різних варіантах управління, що дозволяє змінювати траєкторію зміни станів у процесі функціонування системи і тим самим реалізовувати структурну адаптацію автомата. Елемент безлічі функцій автомата у його стані визначає функцію активації стану, функції виходів і структури. Водночас такий автомат не утворює інтегрованої системи [6].

Моделі операційних автоматів класифікують за місцезнаходженням у системі по відношенню до керуючого автомата (вхідні та вихідні) та по виду виконуваної операції [23]. Вхідний операційний автомат формує вхід (елемент множини входів) керуючого автомата. Логіка виконуваної у своїй операції визначає сенс активності входу. Наприклад, активність входу «Температура більша за норму» вимагає різних керуючих дій (виходів) при різних нормах. Таким чином, змінюючи параметр «норма» операційного автомата, ми впливаємо на логіку управління. Вихідний операційний автомат формує вплив на об'єкт управління виходячи з активності виходу (елемента безлічі виходів) управляючого автомата. Параметри цього впливу можуть бути об'єктом управління з боку вищого керуючого автомата.

Модель елемента інтегрованої системи уявимо як таку, що складається з двох взаємопов'язаних автоматів СА1 та СА2 у комплексі з вхідними ВхОА1, ВхОА2 та вихідним ВихОА1 операційними автоматами [7].

Елемент представляє дворівневу інтегровану систему. У цій системі автомат СА1 у підсистемі нижнього рівня є керуючим автоматом, а в підсистемі верхнього рівня – об'єктом управління з боку автомата СА2. Автомат СА1 отримує події X1 від вхідного операційного автомата ІОА1 та варіант управління С1 від вищого автомата СА2. У свою чергу, автомат СА1 формує сигнали управління Y01 вихідним операційним автоматом ООА та інформаційні сигнали Y12 для автомата СА2. Автомат СА2 отримує події X2 від автомата ІОА2 та варіант управління С2 від вищого рівня ієрархічної системи. Крім цього, автомат СА2 формує сигнали управління С1 та інформаційні сигнали Y23 [7].

Під керованим автоматом розумітимемо автомат CA1, у якого, за допомогою іншого автомата CA2, можна змінити хоча б один елемент його кортежу. Наприклад, блокувати частину входів [7], змінити початковий стан та інші. Можлива одночасна зміна кількох і навіть усіх елементів кортежу. Такі зміни, зазвичай, призведуть до зміни результатів функціонування автомата. Автомат CA2, який змінює елементи кортежу автомата CA1, назвемо керуючим. У загальному випадку, керований автомат може отримувати керування від більш ніж одного керуючого автомата. І навпаки, керуючий автомат може керувати більш ніж одним автоматом [7].

Сигнал управління, сформований керуючим автоматом CA2, може ініціювати зміни у вхідних операційних автоматах. Наприклад, зміни порогів сигналів, що формують події (входи) автомата. Метою таких змін може бути параметрична адаптація пристрою керування системою [7].

Багато входів автомата характеризує спостереження процесів в об'єкті управління. Сенс блокування окремих входів автомата у певному стані у тому, щоб змінити кількість виконуваних циклів управління у автоматі (можливих шляхів у графі автомата), отже, і функціональні можливості управління. Зміна початкового стану автомата може скоротити час виходу системи раціональний цикл управління [7].

Безліч виходів автомата характеризує керованість об'єкта. Для автомата нижнього рівня це об'єкт системи, а інших автоматів – нижче лежачі операційний і управляючий автомат. У бінарних автоматах управління виходами зводиться до їх блокування/розблокування, в небінарних, а також – до зміни параметрів впливів (рівень, номенклатура, часовий інтервал) [6], [29].

Блокування активності стану автомата еквівалентна блокуванню всіх входів, які призводять до переходу в цей стан та блокування дій у ньому [7].

Таким чином, управління елементами множин кортежу автомата змінює його функції у межах. Суб'єктом такого управління є автомат CA2 (рис. 2.9). Виходи цього керуючого автомата в його активному стані визначають вектор управлінь автоматом CA1, у тому числі початковий стан CA1, діяльність

в активному стані та шляхи переходу з цього стану в інші. Елементи вектору управлінь можуть бути дискретного (дозволити/блокувати елемент безлічі кортежу керованого автомата) або аналогового типу. В останньому випадку значення елемента управління можна розглядати, наприклад, як поріг для формування вхідним операційним автоматом події входу керованого автомата або як коефіцієнт, на який множиться амплітуда вихідного сигналу вихідного операційного автомата або як тривалість активності виходу або як бінарний код, одиничні виходи в якому служать дозволом окремих виходів підмножини виходів керованого автомата у цьому стані [6], [7].

Структура керуючого автомата СА2 задає траєкторію зміни управлінь СА1 в рамках деякого циклу управління об'єктом. Ці траєкторії, своєю чергою, може бути об'єктом управління наступному рівні ієрархії управлінь у системі. Типові траєкторії зміни управлінь забезпечують розширення/звуження функціональних можливостей, зменшення/збільшення числа етапів у циклах управління, зменшення/збільшення числа незаблокованих входів автомата нижнього рівня [7].

Структурну схему модернізованого типового елемента інтегрованої системи наведено на рис. 2.9 [6].

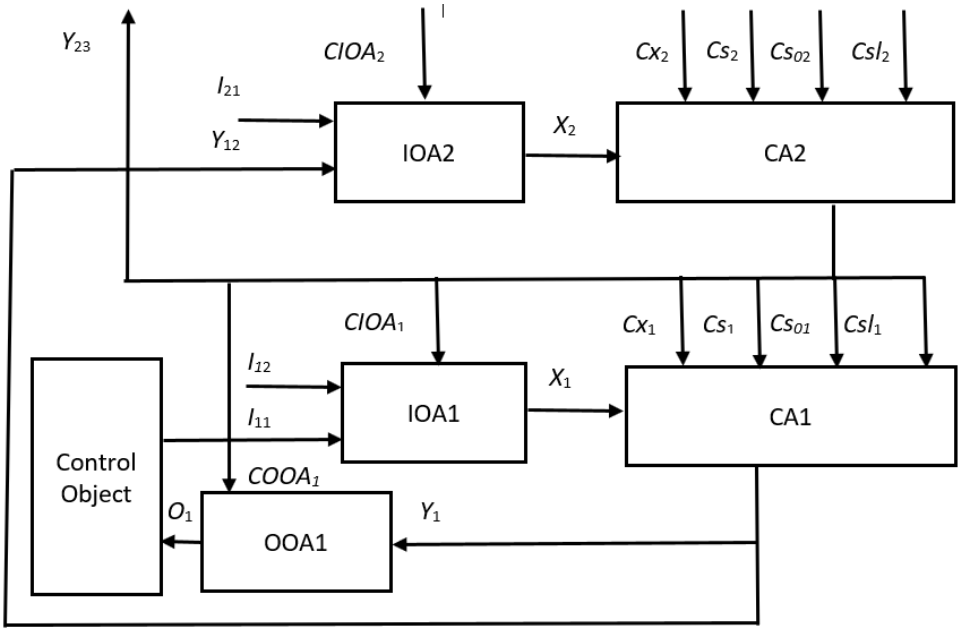


Рисунок 2.9 – Структурна схема типового елемента інтегрованої системи

На схемі показані керуючі автомати першого (СА1) та другого (СА2) рівнів, вхідні операційні автомати першого (ІОА1) та другого (ІОА2) рівнів, вихідний операційний автомат першого (ООА1) та об'єкт управління системи. Об'єкт управління системи реагує впливу О1 зміною параметрів І11. Автомат ІОА1 використовує параметри об'єкта І11 та параметри зовнішнього середовища І12 для формування входів автомата Х1 СА1. У свою чергу автомат під дією входів Х1 змінює свій стан і формує виходи впливів Y1 для автомата ООА1 і інформаційні виходи Y12 для інформування автомата СА2. Вхід СІОА1 управляє параметрами автомата ІОА1, вхід СООА1 – параметрами автомата ООА1. А входи Сх1, Cs1, Cs01, Csl1 управляють структурою автомата СА1. Всі перелічені сигнали управління формуються як виходи СА2, який також має аналогічне управління від вище лежачого автомата [6].

Вибір POU для реалізації елементів ТЕІС мовами програмування промислових контролерів зроблено виходячи з таких міркувань [6]:

- на входи ІОА надходять результати вимірювання параметрів об'єкта. Автомат ІОА, як правило, виконує операцію перетворення, обчислення математичних виразів, перевірку здійсненності логічних умов. У цьому випадку автомат ІОА може бути реалізований як функція або функціональний блок мовами програмування ST або FBD. А сигнал керування СІОА буде мати сенс масштабуючого коефіцієнта або елемента логічної умови. Наприклад, для автомата ІОА, який формує вхід х1 "Температура в нормі" управління полягає у зміні параметрів цієї норми;
- керуючі автомати у програмі управління реалізовані як функціональні блоки мовою SFC: стани автомата відповідають крокам програми; входи автомата – умов переходів; кільком варіантам виходу зі стану відповідають альтернативні гілки;
- логіка Сх1 управління входами реалізується лише на рівні POU «програма». При цьому кожен вхід пристрою, що управляє, надходить на відповідний вхід автомата тільки за наявності дозволу від логіки Сх1, яка реалізується за допомогою POU функціонального блоку;

- логіка Cs1 блокування стану зводиться до блокування входів всіх можливих переходів у цей стан;
- для реалізації логіки Cs01 зміни початкового стану, в автомат, що управляє, вноситься додаткові входи для переходу зі стану s0 в інший початковий стан, яке задано управлінням Cs01;
- логіка дій може залежати від входу, яким відбулася активація стану автомата. І цією логікою можна керувати задаючи керування Csl1 з боку вищого автомата. Але це суттєво ускладнює програму управління та потребує окремого розгляду;
- автомат ООА реалізується як безліч дій на кожному кроці відповідного функціонального блоку управляючого автомата. Цим діям у кроках програми відповідають виходам автомата. Наприклад, таким як присвоєння вихідним змінним певних числових значень, які можна трактувати як: масштабуючи коефіцієнти амплітуди виходу даного стану; N – розрядний двійковий код, який керує включенням (одиниця в коді) N цифрових виходів пристрою керування; параметр кваліфікатору дії у цьому кроці.

Таким чином, елементи TEIC можуть бути відображені на безліч POU типу програма, функція і функціональний блок і реалізовані мовами програмування стандарту IEC 61131-3.

3 МЕТОДИКА ПРОТОТИПУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕІС

3.1 Методика прототипування

Під прототипом зосередженої системи управління на основі PLC розумітимемо її функціональний аналог, в якому PLC заміщений спрощеною мікроконтролерною платою, при цьому [30]:

- вихідний код керуючої програми, яка виконується в мікроконтролері плати, написаний мовами програмування за стандартом IEC 61131-3 та функціонально повторює програмний код системи оригіналу;
- до входів та виходів мікроконтролерної плати підключено всі датчики та виконавчі механізми об'єкта керування, які підключені до PLC у реальній системі. У прототипі системи сигнали від датчиків та до виконавчих механізмів узгоджені за рівнями та полярністю з каналами мікроконтролерної плати;
- є таблиця відповідності входів/виходів PLC, каналів мікроконтролерної плати та їх адреси у програмі;
- тимчасові константи у програмі мікроконтролерної плати залежні від тактової частоти процесора узгоджені з константами керуючої програми PLC;
- зв'язку мікроконтролерної плати з персональним комп'ютером через віртуальний COM-порт достатньо для перевірки функціональності системи за допомогою її прототипу.

У разі невиконання хоча б однієї з перелічених вимог, говоритимемо про не повну функціональну відповідність прототипу оригіналу. Але, у багатьох випадках, це також корисно, наприклад при навчанні програмуванню PLC. Можливо два варіанти використання прототипування системи керування [30], [31]:

- спроєктовано систему керування на базі PLC, але обраний PLC фізично не доступний. У цьому випадку прототипування спрощує перевірку правильності алгоритму керування реальним об'єктом чи його макетом;
- спроєктований та протестований прототип системи керування на базі мікроконтролерної плати та макета об'єкта керування. Ця робота спрощує вибір PLC.

Вихідними даними для прототипування системи керування є електрична схема системи та моделі операційних та керуючого автоматів, які треба реалізувати у програмно-апаратному комплексі прототипу [30].

На першому етапі проводиться вибір мікроконтролерної плати з урахуванням необхідної кількості цифрових та аналогових каналів введення/виводу, каналів з ШІМ, оцінюється потреба у додаткових вузлах, які відсутні у платі прототипу. Це можуть бути, наприклад, вузли узгодження входів/виходів прототипу за рівнем та потужністю, додаткові джерела живлення. Багато PLC працюють у діапазоні напруги 24 В, а мікроконтролерні плати – у діапазоні 5 В [30].

На другому етапі проводиться вибір структури та елементів POU для проєкту прототипу керуючої програми. Програма є типом POU на верхньому рівні організації проєкту прототипу. У проєкті може бути кілька програм. Операційні автомати реалізуються як функції чи функціональні блоки, а управляючі – як функціональні блоки. Для зв'язку компонентів програми з апаратною частиною мікроконтролерної плати використовуються змінні класів «вхід» та «вихід». Функціональні блоки описуються у програмі як локальні класи [30].

На третьому етапі обираються мови програмування для POU проєкту прототипу програми, що управляє. Різні мови стандарту IEC 61131-3 мають різні ефективності при реалізації компонентів проєкту прототипу системи керування. При виборі мови слід перевірити її доступність у планованому PLC. Якщо є можливість вибору, то реалізації ОА вибираються ефективні мови ST, FBD, рідше – мова IL. У той же час, КА простіше реалізувати мовою SFC, але

можливо і іншими мовами. Таким чином, типова керуюча програма для прототипу системи є мультимовною [30].

Розглянемо приклад прототипування найпростішої дискретної системи стабілізації температури об'єкта. Об'єкт системи містить датчик температури та нагрівач. За відсутності нагрівання об'єкт остигає, передаючи теплову енергію у зовнішнє середовище. Зовнішнє середовище формує завдання I1 на температуру об'єкта. Операційний автомат ВхОА порівнює фактичну I2 і задану I1 температури та формує події (входи X): «температура менша за завдання», «температура більша за завдання». Ці входи визначають поточний стан керуючого автомата («нагрів» або «охолодження»), в якому формується вихід Y. Операційний автомат ВихОА перетворює цей вихід сигнал керування нагрівачем.

Проєкт Prototype включає функціональні блоки ОА і КА, які виконуються в програмі LimTemp [30]. Ця програма реалізує алгоритм керування нагрівом об'єкта системи за якого нагрівання відбувається за умови, що температура об'єкта менша за граничну. Графічне уявлення функціональних блоків ОА, КА та програми LimTemp наведено на рис.3.1 – 3.4, відповідно.

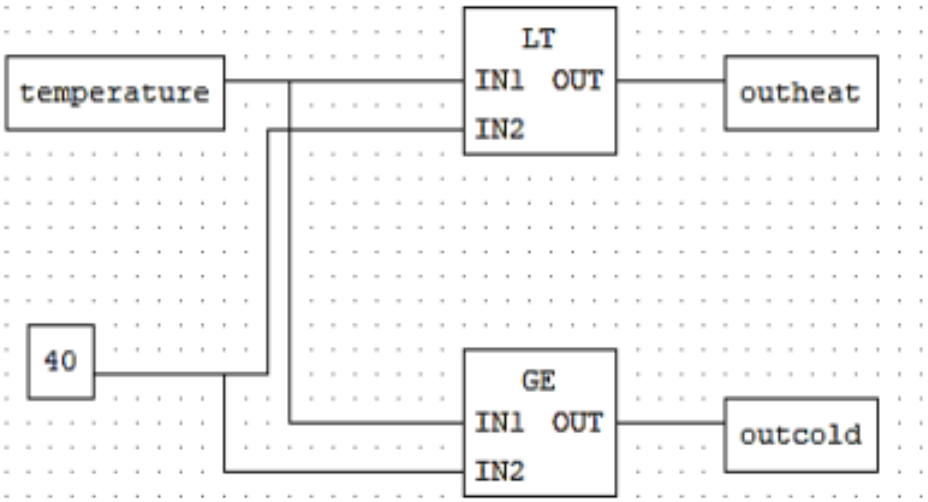


Рисунок 3.1 – Графічне уявлення мовою FBD функціонального блока ОА програми LimTemp

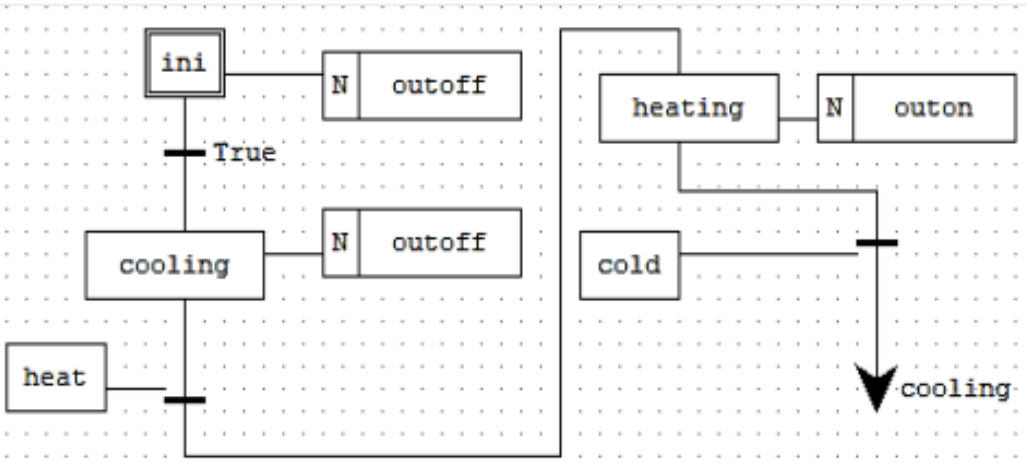


Рисунок 3.2 – Графічне уявлення мовою FBD функціонального блока КА програми LimTemp

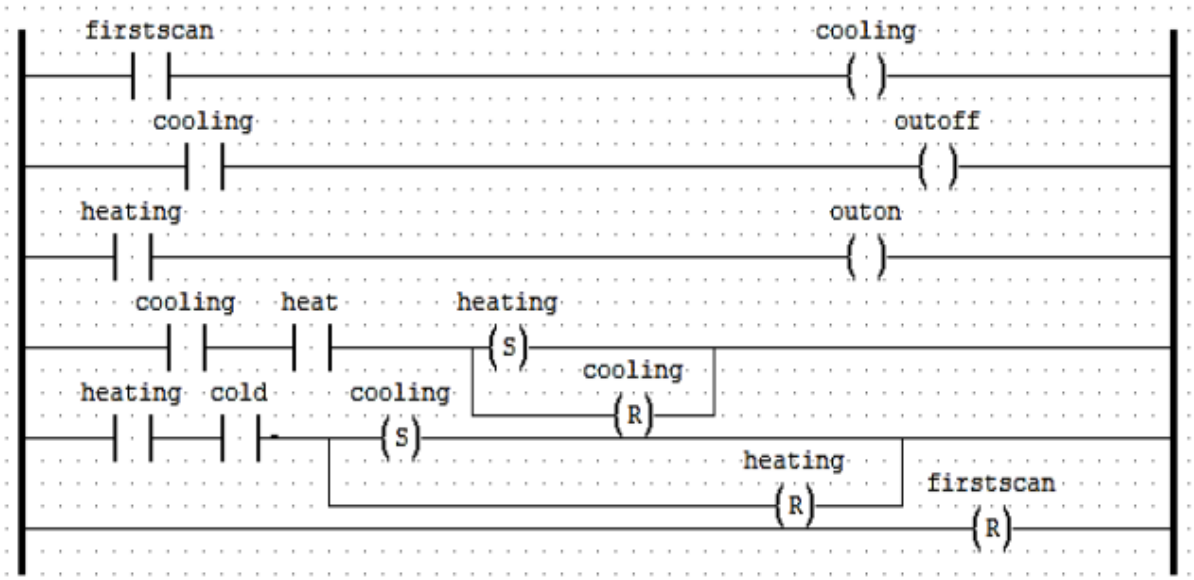


Рисунок 3.3 – Графічне уявлення мовою LD функціонального блока КА програми LimTemp

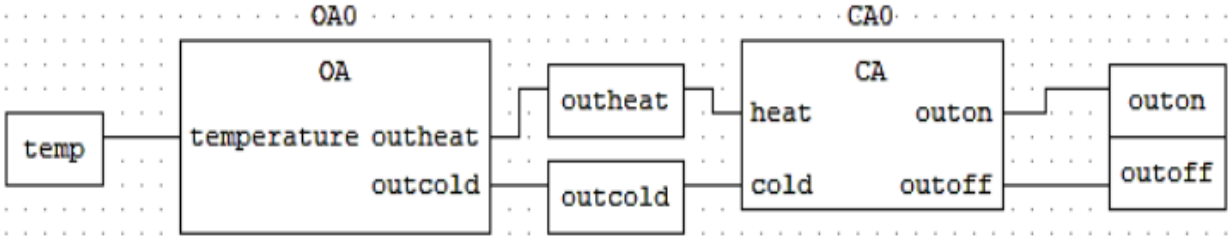


Рисунок 3.4 – Графічне уявлення мовою FBD програми LimTemp

Перевірка проєкту Prototype у середовищі додатку OpenPLC Editor з використанням логічного PLC для обох версій автомата КА підтвердила відповідність проєкту вимогам системи керування [30] – [32]. Бажане значення змінної temp встановлювалося у вікні налагоджувача "форсувати значення змінної".

Перевірка проєкту Prototype із виконанням програмного коду в платі Ардуїно, до якої підключено прототипи датчика температури та нагрівача проводилася з використання програм OpenPLC Editor та OpenPLC Runtime. Фактична температура задавалася шляхом нагрівання датчика. При цьому сигнали керування нагріванням підтверджувались світлодіодами і відповідали вимогам системи керування.

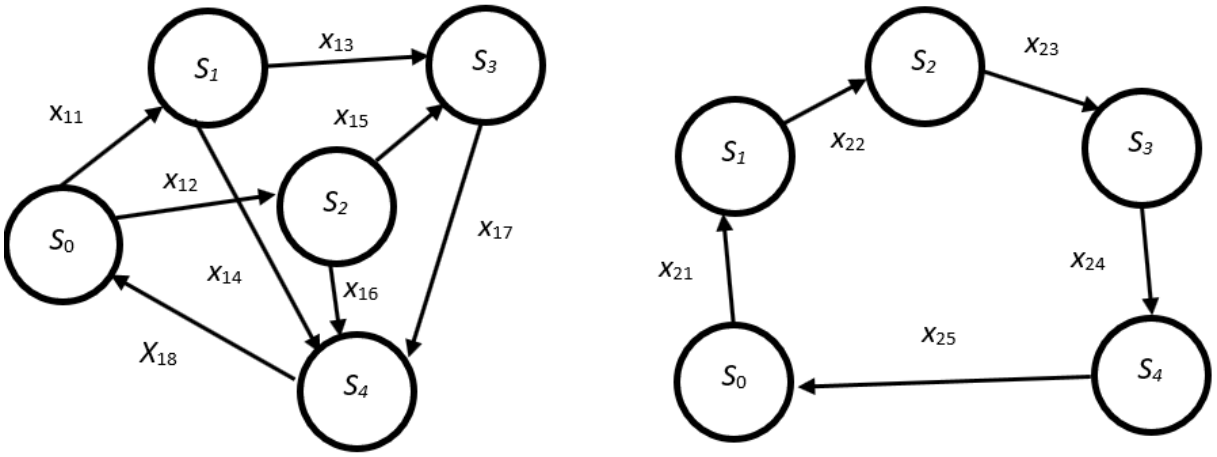
3.2. Програмна реалізація та тестування елемента інтегрованої системи

Розглянемо реалізацію елемента інтегрованої системи на прикладі автоматів КА1 і КА2, заданих графами наведеними на рис. 3.5. [6] Автомат КА1 має безліч станів $S = (s0, s1, s2, s3, s4)$, входів $X = (x1,x2, x3, x4, x5, x6, x7, x8)$ і виходів $Y=(y0, y1, y2, y3, y4)$. Причому вихід $y0$ активний у стані $s0$, вихід $y1$ – у стані $s1$ тощо. Як видно із рис. 3.5а через вершину s0 в автоматі КА1 проходять шляхи чотирьох циклів управління об'єктом системи. Це шлях 1: s0-s1-s3-s4-s0; шлях 2: s0-s1-s4-s0; шлях 3: s0-s2-s3-s4-s0 і шлях 4: s0-s2-s4-s0.

Цикл управління змінюється під час переходу активності через вершину s0. Стратегія управління у тому, щоб активність автомата переміщалися в послідовності: шлях 1 – шлях 2 – шлях 3 – шлях 4. Далі послідовність повторюється знову. Стратегію управління формує автомат КА2, в якому є тільки один шлях і зміна стану якого відбувається в момент завершення чергового циклу автомата КА1 [6].

Завдання управління автоматом КА1 із боку автомата КА2 полягає у обмеженні входів автомата КА1 те щоб він реалізовував лише одне заданий шлях

у поточному циклі управління. Графічне FBD представлення програми елемента інтегрованої системи серед програми OpenPLC наведено на рис. 3.6 [6].



а б

Рисунок 3.5 - Граф керованого КА1 (а) і керуючого КА2 (b) автомата

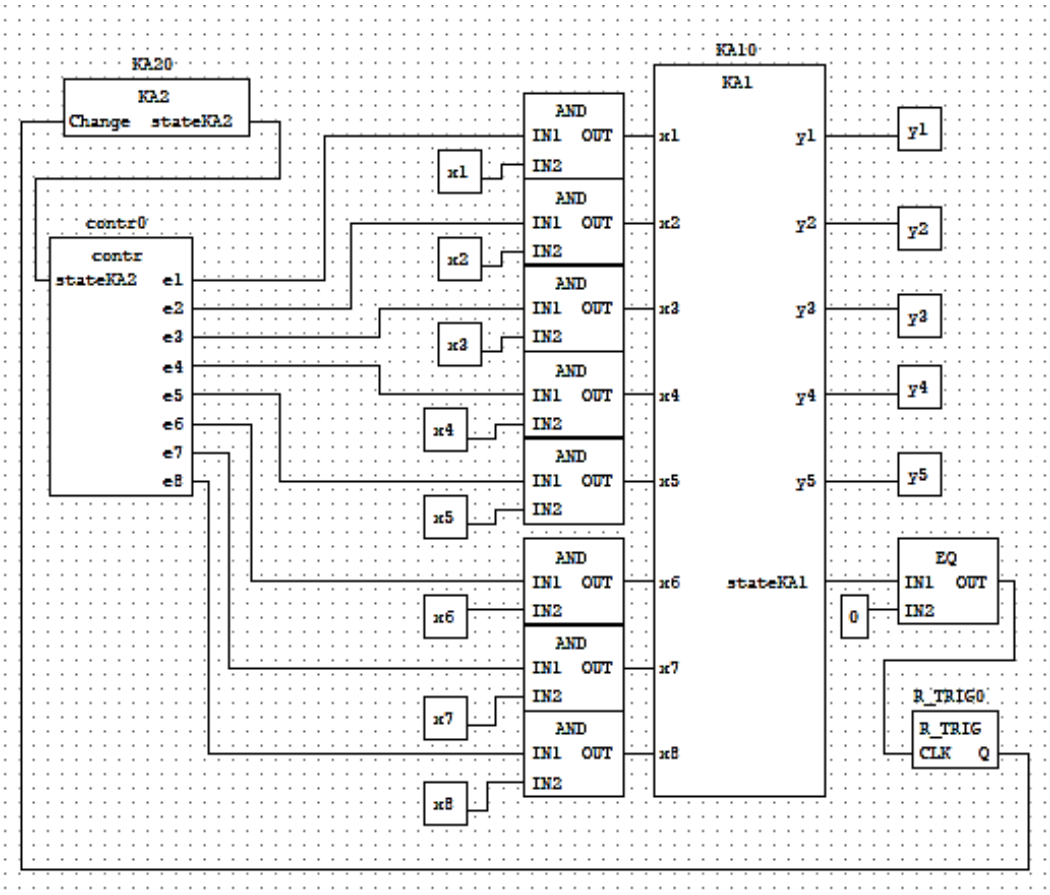


Рисунок 3.6 – Графічне FBD представлення програми елемента інтегрованої системи

Програма містить функціональні блоки типів *KA1*, *KA2* та contr, графічні уявлення яких наведені на рис. 3.7 - 3.9.

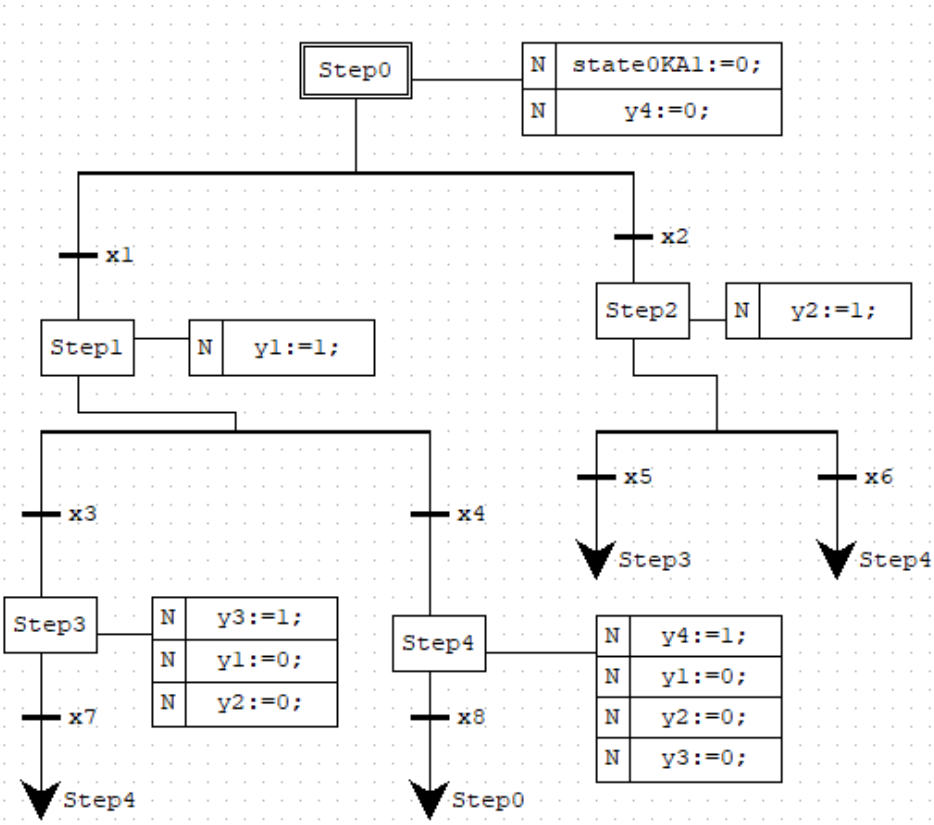


Рисунок 3.7 – SFC функціональний блок автомата КА1

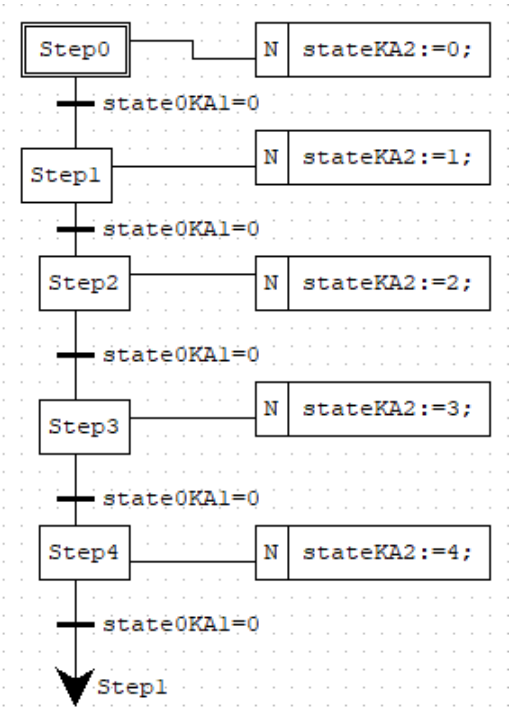


Рисунок 3.8 – SFC функціональний блок автомата КА2

```
CASE stateKA2 OF
   0: e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
   1: e1:=1; e2:=0; e3:=1; e4:=0; e5:=0; e6:=0; e7:=1; e8:=1;
   2: e1:=1; e2:=0; e3:=0; e4:=1; e5:=0; e6:=0; e7:=0; e8:=1;
   3: e1:=0; e2:=1; e3:=0; e4:=0; e5:=1; e6:=0; e7:=1; e8:=1;
   4: e1:=0; e2:=1; e3:=0; e4:=0; e5:=0; e6:=1; e7:=0; e8:=1;
   ELSE
     e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
     END_CASE;
```

Рисунок 3.9 – ST функціональний блок *contr* керування дозволами входів автомата КА1

У початковий момент обидва автомати встановлюються стан *s*0 (змінні stateCA1 = 0 і stateCA2=0), на вхід Change автомата КА2 приходить імпульс, який встановлює автомат КА2 в стан *S*1. Внаслідок цього блок count формує дозволи/заборони на входах *e*1 – *e*8, які відповідають стратегії «1». Дозволені події з множини *x*1 - *x*8 надходять на входи автомата КА1 і викликають переходи цього автомата в інші стани за циклом управління об'єктом системи. У момент завершення цього циклу на вході Change формується новий імпульс і описаний цикл повторюється для стратегій "2", "3", "4", "0", "1", "2". [6]

Наведений на рис. 3.7 SFC функціональний блок автомата КА1 описує структуру переходів та дії в станах згідно з графом автомата рис. 3.5а. [6]

Аналогічні завдання вирішує SFC функціональний блок автомата КА2, наведений на рис. 3.8. У даному прикладі автомат СА2 є некерованим, але якщо в системі з'явиться ще один рівень управління, цей автомат може бути об'єктом управління з боку вище лежачого рівня [6].

ST функціональний блок contr (рис. 3.9) управляє дозволами входів автомата КА1 в описаних вище стратегіях управління. З додаванням нової стратегії слід додати новий елемент оператора CASE. Наприклад, якщо дозволити всі входи, то керування може виконуватися за будь-яким із описаних вище циклів залежно від послідовності появи подій на входах *x*1 – *x*8 [6].

На рис. 3.10 наведено приклад SFC – моделі керованого автомата КА2 із двома стратегіями управління рівня КА2 [6].

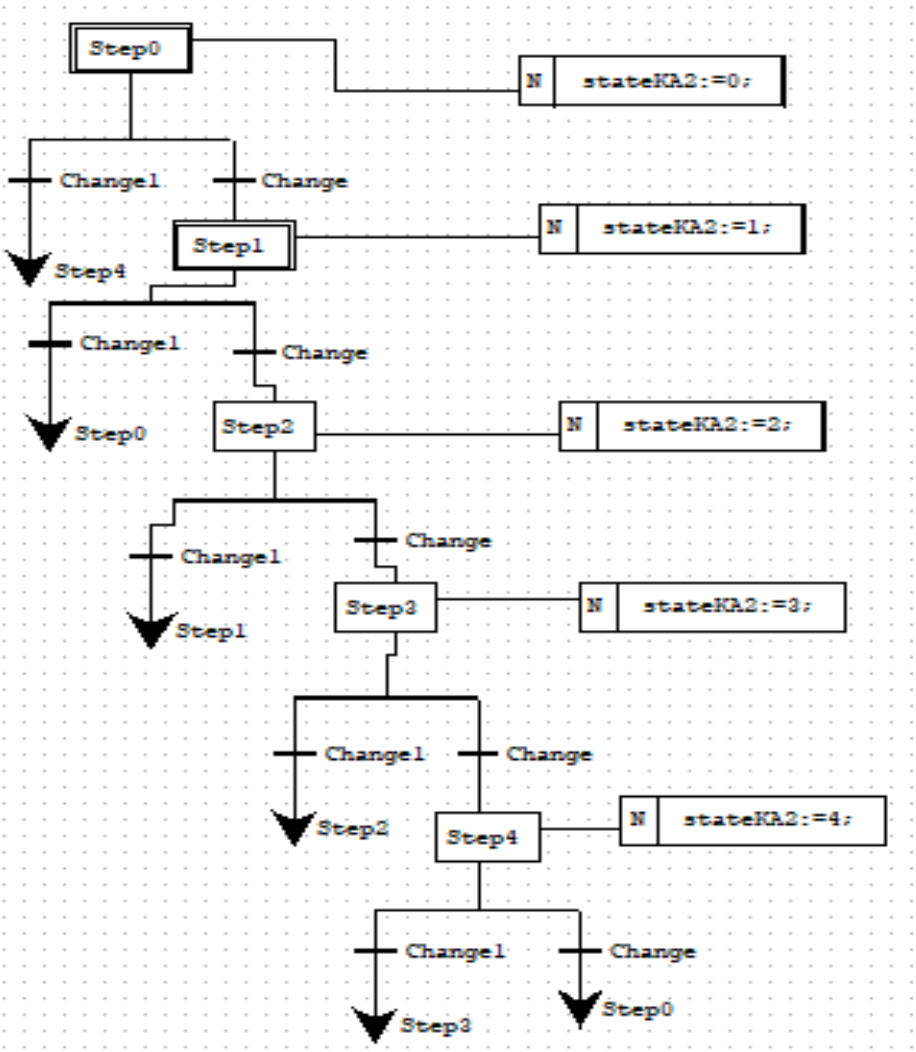


Рисунок 3.10 – SFC модель керованого автомата КА2 із двома стратегіями управління

У першій стратегії рівня КА2 зміна стратегій для управління автоматом КА1 проводиться у порядку: шлях 1 - шлях 2 - шлях 3 - шлях 4. А в другій стратегії рівня КА2 – у зворотному порядку. Для першої стратегії використовується роздільна здатність входу *Change*, а для другої – *Change1*. Ці сигнали формуються на виходах автомата КА3 третього рівня ієрархії управлінь системи [6].

3.3. Аналіз результатів тестування

Моделювання адаптивної поведінки доцільно має відображати взаємозв'язок двох сусідніх рівнів управління в ієрархічній системі. Об'єктом адаптації може бути практично будь-який елемент кортежу управляючих автоматів і параметр операції виконуваної операційними автоматами системи [6].

Експерименти на прикладі показали практичну цінність запропонованої методики моделювання адаптивного керування в інтегрованій ієрархічній системі. Модель використовує виразні засоби графічних мов програмування промислових контролерів. POU цієї моделі мають два рівні. Операційні та управляючі автомати TEIC представлені на нижньому рівні у вигляді функцій або функціональних блоків. Другий рівень – рівень програми відображає логіку взаємозв'язку цих елементів [6].

Графічне представлення моделі в додатку OpenPLC описує логіку управління в системі на функціональному рівні, одночасно допускає трансляцію в код і трансфер цього коду в широку гамму популярних мікроконтролерних плат [6].

Моделювання інтегрованих систем у додатку OpenPLC в рамках навчального процесу підготовки магістрів дозволяє вивчати програмування промислових контролерів з використанням вільно розповсюджуваного програмного забезпечення недорогих популярних мікроконтролерних плат у складі портативної лабораторії для вивчення електроніки, мікроконтролерної техніки та промислових контролерів [25, [32].

4. НАПРЯМКИ ВИКОРИСТАННЯ ТЕІС

4.1. Визначення вищих форм знань IoT системи

У наведених прикладах керовані автомати та ТЕІС використовуються для побудови інформаційно-керуючих систем. Тобто знання у цих системах існують у формах даних та інформації. При цьому використання ТЕІС забезпечує підвищену адаптивність системи та зниження трудомісткості проєктування системи.

Але розвиток IoT систем відбувається у напрямку використання вищих форм знань – знань, розуміння та мудрості. Використання цих форм знань недостатньо розкрито у відомих друкованих виданнях. Тому запропоновані визначення, одиниці, моделі, операції з цими знаннями, а також відношення знань до когнітивних властивостей системи які наведені у табл. 4.1 - 4.3 [10].

Форма «знання» відрізняється від форми «інформація» тим, що містить результат оцінки інформації на відповідність певним умовам або вимогам [10].

Форма «розуміння» відрізняється від форми «знання» тим, що встановлює зв'язок між знаннями та досвідом отримання результатів функціонування системи/підсистеми. Наприклад, зв'язок з її продуктивністю, ступенем досягнення операційної мети. Оскільки підсистема діяльності CCS є ієрархічною, розуміння описується як досвід, отриманий та використаний на різних рівнях цієї ієрархії. Типові операції зі знаннями цієї форми вимагають використання когнітивних здібностей планування, навчання та міркування [10].

Форма «мудрість» відрізняється від форми «розуміння» тим, що встановлює зв'язок між набором розумінь та вибором цілей/сценаріїв/керуючих автоматів/причинно-наслідкових відносин для системи у стані керуючого автомата. Тобто вибір здійснюється на основі набору розумінь, збалансовано та з прогнозуванням у ширших часових рамках [10].

Таблиця 4.1 – Характеристики форми «знання»

Форма знання	Одиниця знання	Операція зі знаннями / когнітивними здібностями	Модель знань
Знання – це результат оцінки інформації, фіксації її відповідності певним умовам або вимогам	Первинний факт: твердження щодо результатів перевірки умови	Оцінка значення тегу / сприйняття	Назва інформації; тип умови, результат перевірки умови
	Вторинний факт: твердження, отримане в результаті логічного висновку з фактів і правил	Логічний висновок / сприйняття	Пролог-програма з фактами, правилами та запитом
	Правило — це n-арний предикат, конструкція для опису відносин між змінними		Предметна область та функція-предикат

Таблиця 4.2 – Характеристики форми «розуміння»

Форма знань	Одиниця знання	Операція на знаннях / когнітивні здібності	Модель знань
1	2	3	4
Розуміння – це зв'язок знань, який описує досвід отримання результатів	Досвід зв'язку знань з цілями функціонування системи	Визначення стану та дій системи з точки зору досягнення мети її функціонування / міркування, навчання, планування	Моделі об'єктів керування, блоків керування (включаючи автомати керування цілями), динаміки вхідних параметрів системи, параметрів для оцінки досягнення цілей.
	Мета полягає в розумінні бажаних результатів системи.		Модель бажаного результату
	Досвід зв'язку знань зі сценаріями	Процес вибору сценарію / міркування	Параметри знань, тип сценарію
	Сценарій – це розуміння того, як досягти конкретної мети.		Модель сценарію – Автомат керування сценарієм

Кінець таблиці 4.2

1	2	3	4
	Досвід зв'язку знань з послідовністю впливів пристрою керування на об’єкт системи	Процес вибору автомата керування об’єктом / міркування	Автомат керування
	Керуючий автомат – розуміння динаміки станів об’єкту в результаті впливів пристрою керування		Модель керуючого автомату
	Досвід зв'язку знань у стані керуючого автомату пристрою керування	Процес вибору методу взаємозв'язку контурів керування та дій / міркування	Семантична машина
	Стан машини – розуміння причинно- наслідкових відносин входів стану з її виходами для обміну знаннями про діяльність та схеми керування.		Моделі циклів станів та керування, а також дій

Таблиця 4.3 – Характеристики форми «мудрість»

Форма знання	Блок знань	Операція на знаннях /когнітивні здібності	Модель знань
Мудрість – це здатність використовувати розуміння	Досвід використання різних розумінь для вибору мети функціонування системи	Процес зміни поточного стану машини цілей / міркування, навчання, планування.	Модель цільової машини
	Досвід використання різних розумінь для вибору методу досягнення мети		Модель сценарної машини
	Досвід використання різних розумінь для вибору послідовності операцій керування блоком керування	Процес зміни поточного стану машини сценаріїв /міркування, навчання, планування	Модель керуючої машини блоку керування
	Досвід використання розумінь для вибору причинно-наслідкових відносин стану автомату	Процес зміни поточного стану машини керування /міркування, навчання, планування	Модель стану [7]

Приклади вищих форм знань наведені в табл. 4.4.

Таблиця 4.4 – Приклади вищих форм знань

Форма знання	Одиниця знань
1	2
Знання	Первинний факт: твердження «температура обмотки нормальна»
	Вторинний факт: твердження «температура обмотки підвищується».
Розуміння	Досвід зв'язку знань з цілями роботи системи: «зниження швидкості зносу допомагає досягти мети збільшення терміну служби трансформатора».
	Мета: продовжити термін служби трансформатора
	Досвід зв'язку знань зі сценарієм: «Зниження температури ізоляції є одним зі сценаріїв зниження швидкості її зносу»
	Сценарій – зниження швидкості зносу ізоляції обмотки
	Досвід зв'язку знань зі станом керуючої машини: якщо температура ізоляції вища за норму, то перехід у стан «Охолодження»
	Керуюча машина – це розуміння динаміки станів автомата під впливом знань про об’єкт керування.
	Досвід зв'язку знань у стані керуючої машини: якщо машина знаходиться в стані «Охолодження», то необхідно ввімкнути пристрій охолодження та дозволити вихід зі стану за подією «температура ізоляції нормальна».
	Стан керуючої машини: «активність охолодження є результатом транзакції (входу в) стан» та «транзакція (виходу зі) стану є результатом охолодження об’єкту керування».
Мудрість	Досвід використання розуміння цілей функціонування системи: якщо метою є забезпечення споживача електроенергією будь-якою ціною, то прискорений знос ізоляції слід ігнорувати.
	Розуміння того, що зниження швидкості зносу ізоляції сприяє збільшенню терміну служби трансформатора, але виходячи з результатів моделювання процесу зносу ізоляції, ми не будемо вживати заходів щодо зниження швидкості зносу (а отже, і навантаження користувача).

Знання, які використовує IoT система зберігаються у базах даних, інформації, знань, розумінь та мудрості. Нові знання надходять до IoT системи у формі сигналів і подальшому конвертуються у вищі форми – від даних до мудрості. Збережені знання використовуються підсистемою діяльності IoT системи, яка реалізована програмно [10].

4.2 Структура ТЕІС вищих форм знань

Функціональну модель підсистеми діяльності опишемо як інтегровану ієрархічну систему, рівні якої взаємодіють за допомогою ТЕІС [15]. У попередньому розділі описаний ТЕІС рівня інформаційно-керуючої системи. У цьому ТЕІС керуючий автомат і-го рівня змінює параметри або структуру керованого автомата (і-1)-го рівня. Розглянемо структуру ТЕІС вищих рівнів до яких віднесемо наступні:

- мети функціонування системи;
- сценаріїв досягнення поточної мети;
- керування поведінкою об’єкту.

На рівні визначення мети функціонування системи ТЕІС складається з керуючого автомата мети (АМ) та керованих їм автоматів сценаріїв (АС) досягнення мети. Керування автоматами АС зводиться до активації одного автомата який відповідає поточній меті [15].

В свою чергу на рівні визначення сценарію досягнення поточної мети ТЕІС складається з автомата сценарію АС та керованих їм автоматів операцій (АО). Керування автоматами АО зводиться до активації одного автомата який відповідає поточному набору операцій [15].

Розглянемо приклад ТЕІС рівня мети функціонування системи керування електрогосподарством. Метою функціонування такої системи можуть бути наступні:

- максимальне збільшення терміну служби обладнання (S1);
- отримання максимального прибутку від експлуатації системи (S2);

- забезпечення електроживлення за будь яких умов (S3);
- запобігання аварії у системі електроживлення. (S4).

Система використовує наступні знання у формі мудрості:

- прибуток важніше за підвищення терміну служби обладнання (x21);
- безперебійність електроживлення важливіша за прибуток (x22);
- безперебійність електроживлення не суттєва (x23);
- важливо запобігти аварії (x24);
- аварія малоймовірна (x25);
- безаварійність важливіша за прибуток (x26).

Перераховані знання мають бінарний тип даних, тобто приймають значення «1» коли вони актуальні та «0» у іншому випадку. Виходи автомату АМ активують відповідний автомат АС.

Граф автомата АМ наведено на рис 4.1.

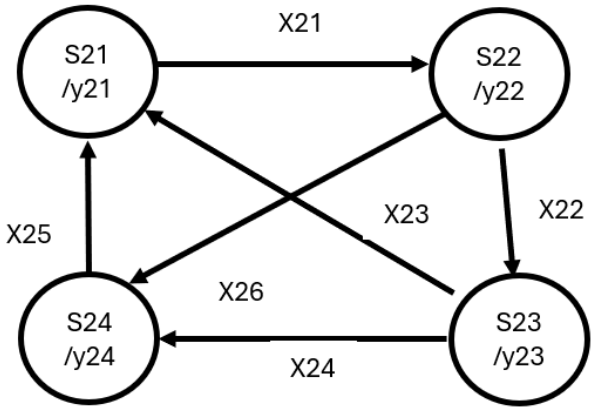


Рисунок 4.1 – Граф автомата мети

Стану S1 автомату АМ відповідає автомат сценарію АС який має наступні стани:

- покращити температурний режим (S11);
- оптимізувати навантаження за часом (S12);
- оптимізувати навантаження за джерелами генерації (S13);
- зменшити (обмежити) навантаження (S14).

Ці стани змінюються під впливом наступних подій:

- охолодження недостатньо для покращення температурного режиму (x11);
- при оптимальному розкладі є прискорена витрата експлуатаційного ресурсу (x12);
- при повному використанні генеруючої потужності є прискорена витрата експлуатаційного ресурсу (x13);
- навантаження на систему істотно впало (X14).

В станах автомата АС ініціюють виходи керування автоматами операцій (АО):

- керування охолодженням (y11);
- розрахунку оптимального розкладу (y12);
- перерозподілу генеруючої потужності (y13);
- формування черг обмеження електроспоживання (y14).

Граф автомата АС максимального збільшення терміну служби обладнання наведено на рис 4.2.

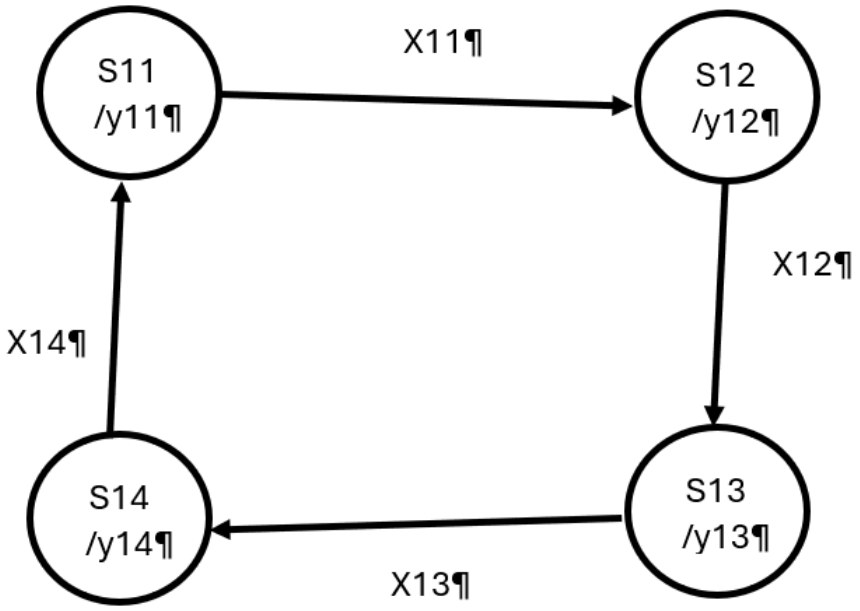


Рисунок 4.2 – Граф автомата сценарію

Структурна схема ТЕІС який поєднує рівні мети та сценаріїв наведено на рис. 4.3.

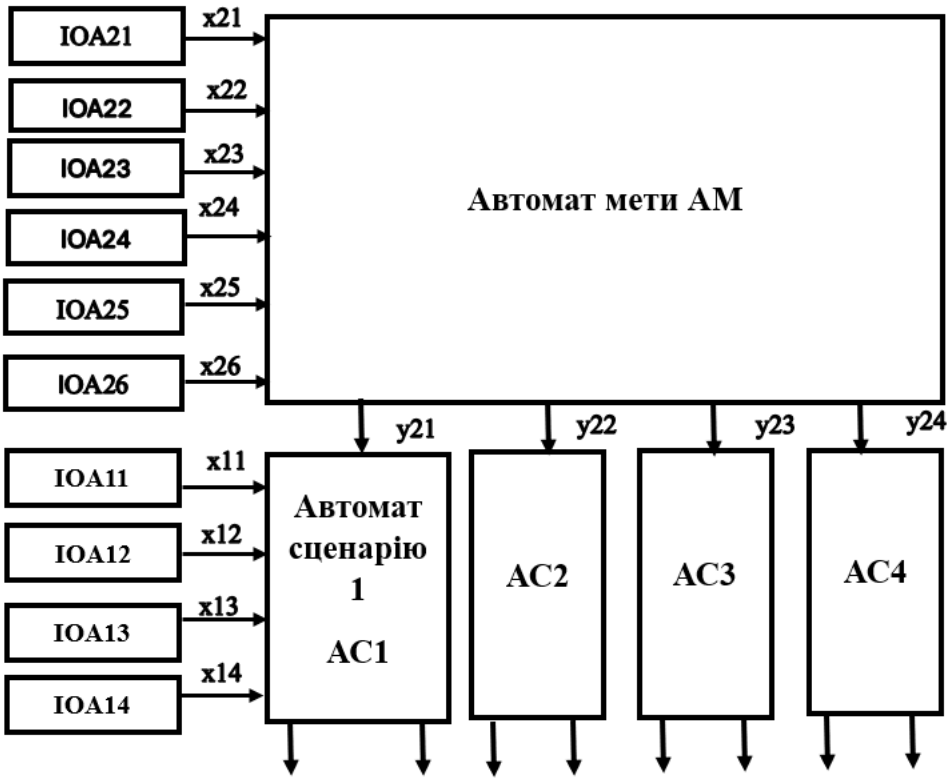


Рисунок 4.3 – Структурна схема ТЕІС який поєднує рівні мети та сценаріїв

Цій елемент ТЕІС сформовано як зв’язка двох та більше керуючих автоматів у який керування автоматами нижчого рівня виконується шляхом їх активації. Входи(події) автоматів АМ, АС1-АС4 Формуються за допомогою вхідних операційних автоматів ІАО. Виходи y21-y24 активують відповідний автомат сценарію АС1-АС4. Графічне відображення елементів POU наведено в додатку Д.

ВИСНОВКИ

IoT є важливим напрямком систем побутової та промислової автоматизації. Він поєднує доступ у реальному часі до важливих параметрів елементів системи з багаторівневою обробкою отриманої інформації.

Проведений аналіз структури складних IoT систем висвітлив переваги побудови таких систем у вигляді ієрархічної інтегрованої структури з використанням формалізму керуючих автоматів та широкими можливостями для системної адаптації. Констатоване також відсутність програмної реалізації типових елементів інтегрованої системи, наявність у її складі PLC.

Для програмної реалізації таких елементів обрані мова програмування C та мови за стандартом IEC 61131-3 а також середовище програмування OpenPLC.

Запропонована методика прототипування дозволяє трансформувати операційні та керуючі автоматів проєктних моделей системи керування у компоненти організації програм проєкту прототипу системи.

Розроблена та протестована методика програмування керуючих автоматів мовами C та мовами стандарту IEC 61131-3. Розроблена структура типового елемента інтегрованої системи, який складається з комплексу керуючого, керованого та операційних автоматів, що забезпечує адаптивну поведінку системи щодо завдань її функціонування.

Визначені типи програмної організації елементів типового елемента інтегрованої системи. Мовами PLC розроблені програми цих елементів. Програми протестовані з використанням середовища програмування OpenPLC та мікроконтролерних плат Arduino. Визначено, що практика застосування типового елемента інтегрованої системи на 10-15 відсотків зменшує трудомісткість програмування завдань адаптивної поведінки складної IoT системи.

Наукова новизна отриманих результатів полягає у тому, що запропоновано структурну модель типового елемента IoT системи, яка складається з двох взаємодіючих керуючих і одночасно керованих скінчених автоматів.

Практична значимість отриманих результатів полягає у тому, що розроблений підхід забезпечує прискорення процесу проєктування програмного забезпечення IoT систем за рахунок використання типових проєктних рішень.

Запропонована методика прототипування пристроїв на основі PLC може бути корисною у навчальному процесі підготовки магістрів.

За темою магістерської роботи опубліковано 8 наукових робіт.

Деякі положення роботи доповідались на Всеукраїнському конкурсі студентських наукових робіт «Інформаційні технології», де робота отримала Диплом ІІІ ступеня.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтернет речей [Електронний ресурс]. – Режим доступу. https://uk.wikipedia.org/wiki/Інтернет_речей.

2. Промисловий_інтернет_речей [Електронний ресурс]. – Режим доступу https://uk.wikipedia.org/wiki/Промисловий_інтернет_речей.

3. Introduction to IoT (Cisco Networking Academy) [Electronic resource]. – Access mode: https://www.netacad.com.

4. The industrial internet of things (IIoT): An analysis framework [Electronic resource] / [H. Boyes, B. Hallaq, J. Cunningham,. et. al;] // Computers in Industry. – 2018.– Volume 101 – P. 1-12. – Access mode : https://doi.org/10.1016/j.compind.2018.04.015.

5. Олещенко Л. М. Програмування пристроїв Інтернету речей : лабораторний практикум : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Програмне забезпечення комп'ютерних та інформаційно-пошукових систем») [Електронний ресурс] / уклад. : Л. М. Олещенко, Я. В. Хіцко. – Електронні текстові дані (1 файл: 2,46 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 47 с.

6. Modeling and Programming of Elements of Integrated Systems in Industrial Controller Languages [Text] / [M. Poliakov, B. Zhurakovskyi,. O. Poliakov et. al.] // CEUR Workshop Proceedings 3702, 2024. – P. 169-178.

7. Poliakov, M. Interoperability of Integrated Hierarchial Systems [Text] / M. O. Poliakov, S. O. Subbotin, O. M. Poliakov // Системні технології : регіональний міжвузівський збірник наукових праць. – 2021. – Вип. 2 (133). – С. 68–78.

8. Poliakov, M. Behavior classification of control unit of systems [Text] / M. Poliakov // Радіоелектроніка, інформатика, управління. – 2022. – № 3. – P. 183-195.

9. Жураковський, Б. Ю. Технології інтернету речей : навч. посіб. для студ. спеціальності 126 «Інформаційні системи та технології», спеціалізація

«Інформаційне забезпечення робототехнічних систем» [Електронний ресурс] / Б. Ю. Жураковський, І. О. Зенів. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 271 с.

10. Poliakov, M. Prospects for studying cognitive control systems in technical universities [Electronic resource] / M. Poliakov, O. Stepanenko, O. Poliakov / International Conference on Academic Studies in Technology and Education (ICASTE), November 12-15, 2025, Antalya, Turkiye : Abstract Book – P. 9. – Access mode : https://www.arste.org/storage/ICASTE/2025/ICASTE2025_Abstract.pdf

11. Frické, M. The Knowledge Pyramid: the DIKW Hierarchy [Text] / M. Frické // IMR Press Knowledge Organization. – January 2019. – 46(1). – P. 33-46 DOI:10.5771/0943-7444-2019-1-33

12. In Cognitive Data Science in Sustainable Computing, Cognitive Computing for Human-Robot Interaction [Electronic resource] / [eds. : M. Mittal, R. R. Shah, S. Roy] // Academic Press. – 2021. – Access mode : https://doi.org/10.1016/B978-0-323-85769-7.00020-3

13. Houwer, D. What is Cognition? A Functional-Cognitive Perspective [Text] / D. Houwer, J. & Barnes-Holmes, Y. & Barnes-Holmes // Core Processes of Cognitive Behavioral Therapies / [eds. : Steven C. Hayes and Stefan G. Hofmann]. – Publisher : Oakland, CA : New Harbinger. – 2016.

14. Schummer, J. From Nano-Convergence to NBIC-Convergence : “The best way to predict the future is to create it” [Text] // J. Schummer // Governing Future Technologies. – Springer Netherlands. – 2009. – P. 57-71. DOI : 10.1007/978-90-481-2834-1_4.

15. Poliakov, M. Structure Of he Course On Studying Functional Extensions of Fsm for Design Hierarchical Integrated Control Systems [Electronic resource] / [M. Poliakov, A. Pirozhok, O. Poliakov et. al.] // International Conference on Studies in Engineering, Science, and Technology (ICSEST) 06 - 10 November 2025, Istanbul, Turkey. – Abstract Book – P. 45.

16. Karpov, Yu. G. Theory of automata [Text] / Yu. G. Karpov. – St. Petersburg: Piter, 2002. – 224 p.

17. Moore, E. F. Gedanken-experiments on sequential machines [Text] / E. F. Moore // Automata Studies, Annals of Mathematics Studies / [eds. : Shannon and McCarthy] – Princeton University Press. – 1956. – Number 34. – P. – 129-153.

18. Mealy, G. H. A method for synthesizing sequential circuits [Text] / G. H. Mealy // The Bell System Technical Journal. – 1955 – Number 34. – P. 1045-1079.

19. IEC 61131-3, Revision 3.0, February 2013 - Programmable controllers – Part 3 : Programming languages [Text] / Published By: International Electrotechnical Commission (IEC). – 2013. – P.468 .

20. Parr, E. A. Programmable Controllers. An engineer's guide [Text] / E. A. Parr]. 3rd ed. – Oxford: Newnes/ – 2003. – P. 429.

21. Hooper, J. F. Introduction to PLCs. Second Edition [Text] / J. F. Hooper // Published By : Carolina Academic Press. – 2006. – P. 120.

22. John, K.-H. IEC 61131-3 : Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision & Making Tools [Text] / K.-H. John // Springer Verlag. – 1995.

23. Віддалений та віртуальний інструментарій в інжинірінгу: монографія / за заг. ред. Карстена Хенке. – Запоріжжя: Дике Поле, 2015. – 250 с.

24. CoDeSys [Electronic resource]. – Access mode : http://www.3s-software.com.

25. OpenPLC Overview – Autonomy (autonomylogic.com) [Electronic resource]. – Access mode : https://autonomylogic.com/d

26. Arduino.cc - guide: windows [Electronic resource]. – Access mode: https://www.arduino.cc/en/Guide/Windows.

27. Instruction Manual "CONTROLLINO" MINI, MAXI and MEGA [Electronic resource]. – Access mode: https://www.controllino.com/tutorials.

28. Поляков, М. А. Конечные автоматы с небинарными элементами множеств [Текст] / М. А. Поляков, И. А. Андриас // Системні технології : регіональний міжвузівський збірник наукових праць. – Дніпро. – 2019. – № 2 (121). – С. 85–94.

29. Поляков, О. М. Прототипування пристроїв керування систем з промисловими контролерами [Текст] / О. М. Поляков, Б.Ю. Жураковський // Системні технології. – Дніпро. – 2024. – 2(151). – С. 50-61. – DOI 10.34185/1562-9945-2-151-202 4-05

30. Поляков, О. М. Методика прототипування пристроїв керування на базі додатку OpenPLC [Текст] / О. М. Поляков, Б. Ю. Жураковський] // In. International scientific and technical conference Information Technologies in Metallurgy and Machine building – ITMM 2024. – Dnipro. – 10-11, April. – 2024. – P. 376-382. – DOI: 10.34185/1991-7848.itmm.2024.01.072.

31. Poliakov, M. O. Principles of construction of the information and control system of the department of the university [Text] / [M. O. Poliakov, P. D. Andrienko, O. M. Poliakov et al.] // System technologies). – Dnipro. – 2023. – N 2(145). – P.30 – 42. – DOI 10.34185/1562-9945-2-145-2023-04.

ДОДАТОК А

Технічне завдання до дипломної кваліфікаційної роботи

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Дослідження та програмна реалізація типових елементів інтегрованих IoT систем» відповідно до наказу № 447 від 30 вересня 2025 року Національного університету «Запорізька політехніка».

А.2 Призначення

Технічне завдання визначає вимоги та вихідні дані до дипломної кваліфікаційної роботи на тему «Дослідження та програмна реалізація типових елементів інтегрованих IoT систем».

А.3 Технічні вимоги

Технічні вимоги:

– провести аналіз структури складних IoT систем, розглянути мови та середовища їх програмування, у тому числі мов програмування за стандартом IEC 61131-3;

– розглянути можливість застосування формалізму керуючих автоматів, способів його програмної реалізації у програмному середовищі програмованих логічних контролерів;

– розробити методику прототипування пристроїв з програмною реалізацією алгоритмів функціонування мовами за стандартом IEC 61131-3. При цьому прототип пристрою побудувати та протестувати на базі мікроконтролерних плат Arduino;

– розробити структуру типового елемента інтегрованої ієрархічної системи для реалізації адаптивної поведінки системи. Для чого використати формалізм автоматів зі зовнішнім керуванням структурою;

– розробити програмну реалізацію типового елемента інтегрованої ієрархічної системи використовуючи вільно розповсюджувані середовища програмування.

А.4. Рекомендована література

Рекомендована література:

– Інтернет речей [Електронний ресурс]. – Режим доступу. https://uk.wikipedia.org/wiki/Інтернет_речей.

– Промисловий_інтернет_речей [Електронний ресурс]. – Режим доступу https://uk.wikipedia.org/wiki/Промисловий_інтернет_речей.

– Introduction to IoT (Cisco Networking Academy) [Electronic resource]. – Access mode: https://www.netacad.com.

– The industrial internet of things (IIoT): An analysis framework [Electronic resource] / [H. Boyes, B. Hallaq, J. Cunningham,. et. al;] // Computers in Industry. – 2018.– Volume 101 – P. 1-12. – Access mode : https://doi.org/10.1016/j.compind.2018.04.015.

– Олещенко Л. М. Програмування пристроїв Інтернету речей : лабораторний практикум : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Програмне забезпечення комп’ютерних та інформаційно-пошукових систем») [Електронний ресурс] / уклад. : Л. М. Олещенко, Я. В. Хіцко. – Електронні текстові дані (1 файл: 2,46 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 47 с.

ДОДАТОК Б

Програма універсального керуючого автомата

```
#include <Arduino.h>

// ================================================
// FINITE STATE MACHINE CONFIGURATION (change only here)
// ================================================

// 1. Initial state number (0-based)
const uint8_t INITIAL_STATE = 0;

// 2. Output table: outputs[state][output_index] = output number (0-based) active in this state
//    Exactly ONE output = 1 per state, others = 0
//    Use {255} if no output is active in this state
const uint8_t OUTPUTS[][1] PROGMEM = {
 {0},  // State 0: output 0 active
 {1},  // State 1: output 1 active
 {2},  // State 2: output 2 active
 {255} // State 3: no active output
};
const uint8_t NUM_STATES = sizeof(OUTPUTS) / sizeof(OUTPUTS[0]);

// 3. Input pin mapping: input_pins[input] = physical Arduino pin
const uint8_t INPUT_PINS[] = {2, 3, 4};  // Input 0 → pin 2, 1 → pin 3, 2 → pin 4
const uint8_t NUM_INPUTS = sizeof(INPUT_PINS);

// 4. Output pin mapping: output_pins[output] = physical Arduino pin
const uint8_t OUTPUT_PINS[] = {10, 11, 12};  // Output 0 → pin 10, 1 → pin 11, 2 → pin 12
const uint8_t NUM_OUTPUTS = sizeof(OUTPUT_PINS);

// 5. Transition table:
```

```
      //    transitions[state][row]:
      //        - bits [0..NUM_INPUTS-1]: allowed input mask (1 =
allowed)
      //      - bits [NUM_INPUTS..NUM_INPUTS+3]: next state number
(0-15)
      //    Each row is one transition rule for the current state.
      //    Rules are checked in order. First match wins.
      //    If no match — stay in current state.
      const uint16_t TRANSITIONS[][4] PROGMEM = {
       // State 0
       { 0b00000001 | (1 << NUM_INPUTS),   // Input 0 → state 1
         0b00000010 | (2 << NUM_INPUTS),   // Input 1 → state 2
         0b00000100 | (0 << NUM_INPUTS),   // Input 2 → stay in 0
         0 },
       // State 1
       { 0b00000001 | (2 << NUM_INPUTS),   // Input 0 → state 2
         0b00000010 | (0 << NUM_INPUTS),   // Input 1 → state 0
         0 },
       // State 2
       { 0b00000001 | (0 << NUM_INPUTS),   // Input 0 → state 0
         0 },
       // State 3 (example placeholder)
       { 0 }
      };
      const uint8_t TRANSITIONS_PER_STATE[] PROGMEM = {4, 3, 2,
1}; // rows per state

      // ================================================
      // INTERNAL VARIABLES
      // ================================================
      uint8_t current_state;
      uint32_t last_input_state = 0;  // for edge detection

      // ================================================
      // HELPER FUNCTIONS
```

```
// ============================================

// Read current state of all inputs → bitmask
uint32_t read_inputs() {
 uint32_t mask = 0;
 for (uint8_t i = 0; i < NUM_INPUTS; ++i) {
   if (digitalRead(INPUT_PINS[i]) == HIGH) {
     mask |= (1UL << i);
   }
 }
 return mask;
}

// Set outputs according to current state
void set_outputs() {
 // Reset all outputs first
 for (uint8_t i = 0; i < NUM_OUTPUTS; ++i) {
   digitalWrite(OUTPUT_PINS[i], LOW);
 }

 // Read active output from table
 uint8_t     active_out     =     pgm_read_byte(&OUTPUTS[cur-
rent_state][0]);
 if (active_out < NUM_OUTPUTS) {
   digitalWrite(OUTPUT_PINS[active_out], HIGH);
 }
}

// Find transition: returns new state or 255 if no transition
uint8_t find_transition(uint8_t state, uint32_t input_mask)
{
 uint8_t         rows         =         pgm_read_byte(&TRANSI-
TIONS_PER_STATE[state]);
 const uint16_t* table = TRANSITIONS[state];

 for (uint8_t r = 0; r < rows; ++r) {
```

```
        uint16_t rule = pgm_read_word(&table[r]);
        if (rule == 0) break;  // end of table

        uint16_t allowed_mask = rule & ((1 << NUM_INPUTS) - 1);
        uint8_t next_state = (rule >> NUM_INPUTS) & 0x0F;

        // Check if current input matches allowed mask and is not
zero
        if ((input_mask & allowed_mask) == input_mask && in-
put_mask != 0) {
          return next_state;
        }
      }
      return 255;  // no transition
    }

    // ================================================
    // SETUP
    // ================================================
    void setup() {
      Serial.begin(9600);

      // Configure input pins
      for (uint8_t i = 0; i < NUM_INPUTS; ++i) {
        pinMode(INPUT_PINS[i], INPUT_PULLUP);  // or INPUT if ex-
ternal pull-ups
      }

      // Configure output pins
      for (uint8_t i = 0; i < NUM_OUTPUTS; ++i) {
        pinMode(OUTPUT_PINS[i], OUTPUT);
        digitalWrite(OUTPUT_PINS[i], LOW);
      }

      // Initialize FSM
      current_state = INITIAL_STATE;
```

```
  set_outputs();

  Serial.print(F("FSM started. Initial state: "));
  Serial.println(current_state);
 }

 // ================================================
 // LOOP
 // ================================================
 void loop() {
  uint32_t current_inputs = read_inputs();

  // Detect change (edge)
  uint32_t changed = current_inputs ^ last_input_state;
  if (changed == 0) {
    delay(10);
    return;  // no change
  }

  // Detect which input(s) went HIGH
  uint32_t triggered = current_inputs & changed;
  if (triggered == 0) {
    last_input_state = current_inputs;
    return;  // only falling edge — ignore
  }

  // Find transition
  uint8_t next_state = find_transition(current_state, trig-
gered);
  if (next_state != 255) {
    current_state = next_state;
    set_outputs();

    Serial.print(F("Transition: "));
    Serial.print(current_state);
    Serial.print(F(" ← input "));
```

```
    for (uint8_t i = 0; i < NUM_INPUTS; ++i) {
      if (triggered & (1UL << i)) {
        Serial.print(i);
        Serial.print(' ');
      }
    }
    Serial.println();
  }

  last_input_state = current_inputs;
  delay(50);  // simple debounce
}
```

ДОДАТОК В

Програмна реалізація ТЕІС

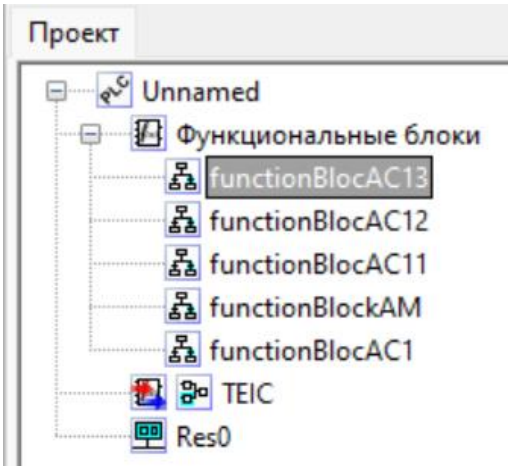


Рисунок В.1 – Структура проєкту TEIC

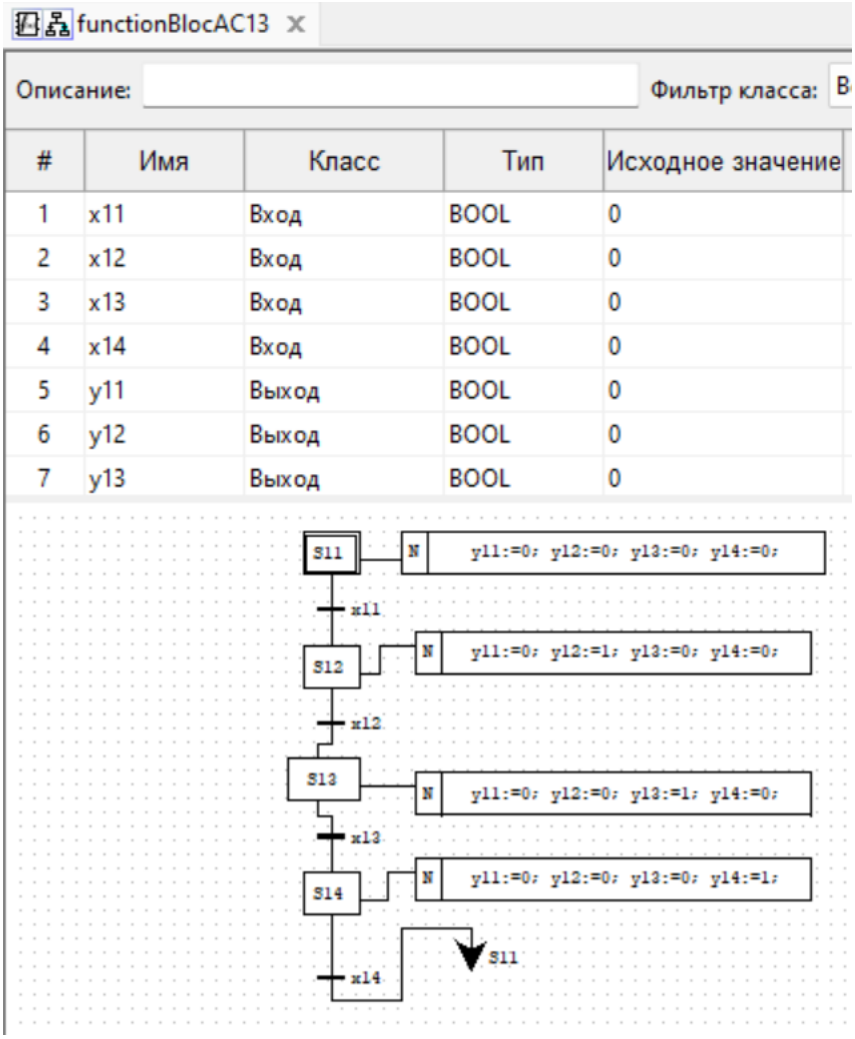


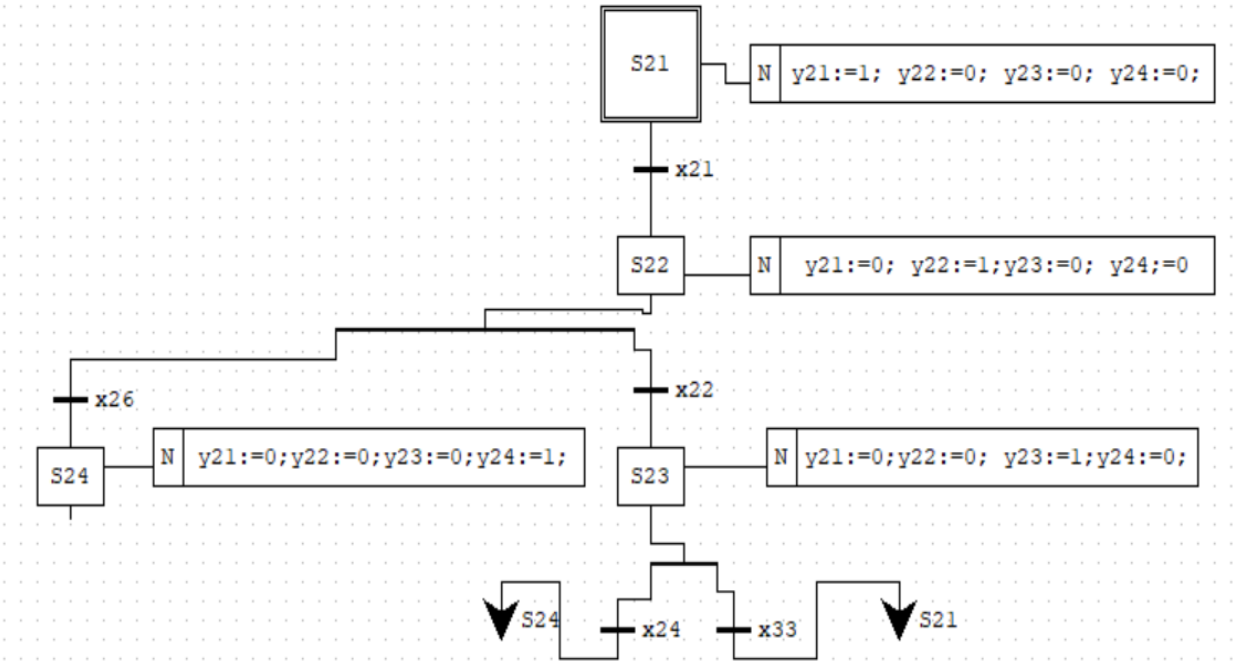
Рисунок В.2– SFC представлення програми автомата сценарію AC1

functionBlocAC13 functionBlockAM

Описание: Фильтр класса:

#	Имя	Класс	Тип	Исходное значение
1	x21	Вход	BOOL	
2	x22	Вход	BOOL	0
3	x23	Вход	BOOL	0
4	x24	Вход	BOOL	0
5	x25	Вход	BOOL	0
6	x26	Вход	BOOL	0
7	y21	Выход	BOOL	0
8	y22	Выход	BOOL	0
9	y23	Выход	BOOL	0
10	y24	Выход	BOOL	0

а)



б)

а – опис класів; б – SFC представлення

Рисунок В.3 – Програма автомата мети

Описание:

#	Имя	Класс	Тип
1	x21	Вход	BOOL
2	x22	Вход	BOOL
3	x23	Вход	BOOL
4	x24	Вход	BOOL
5	x25	Вход	BOOL
6	x26	Вход	BOOL
7	c1x11	Вход	BOOL
8	c1x12	Вход	BOOL
9	c1x13	Вход	BOOL
10	c1x14	Вход	BOOL
11	c2x11	Вход	BOOL
12	c2x12	Вход	BOOL
13	c2x13	Вход	BOOL
14	c2x14	Вход	BOOL
15	c3x11	Вход	BOOL
16	c3x12	Вход	BOOL
17	c3x13	Вход	BOOL
18	c3x14	Вход	BOOL
19	c4x11	Вход	BOOL
20	c4x12	Вход	BOOL

Рисунок В.4 – Опис класів програми TEIC частина 1

Описание:

#	Имя	Класс	Тип
19	c4x11	Вход	BOOL
20	c4x12	Вход	BOOL
21	c4x13	Вход	BOOL
22	c4x14	Вход	BOOL
23	y21	Выход	BOOL
24	y22	Выход	BOOL
25	y23	Выход	BOOL
26	y24	Выход	BOOL
27	c1y11	Выход	BOOL
28	c1y12	Выход	BOOL
29	c1y13	Выход	BOOL
30	c1y14	Выход	BOOL
31	c2y11	Выход	BOOL
32	c2y12	Выход	BOOL
33	c2y13	Выход	BOOL
34	c2y14	Выход	BOOL
35	c3y11	Выход	BOOL
36	c3y12	Выход	BOOL
37	c3y13	Выход	BOOL
38	c3y14	Выход	BOOL
39	c4y11	Выход	BOOL
40	c4y12	Выход	BOOL
41	c4y13	Выход	BOOL
42	c4y14	Выход	BOOL
43	functionBlockA	Локальный	functionBlockAM
44	functionBlocAC	Локальный	functionBlocAC11
45	functionBlocAC	Локальный	functionBlocAC11
46	functionBlocAC	Локальный	functionBlocAC11
47	functionBlocAC	Локальный	functionBlocAC11

Рисунок В.5 – Опис класів програми TEIC частина 2

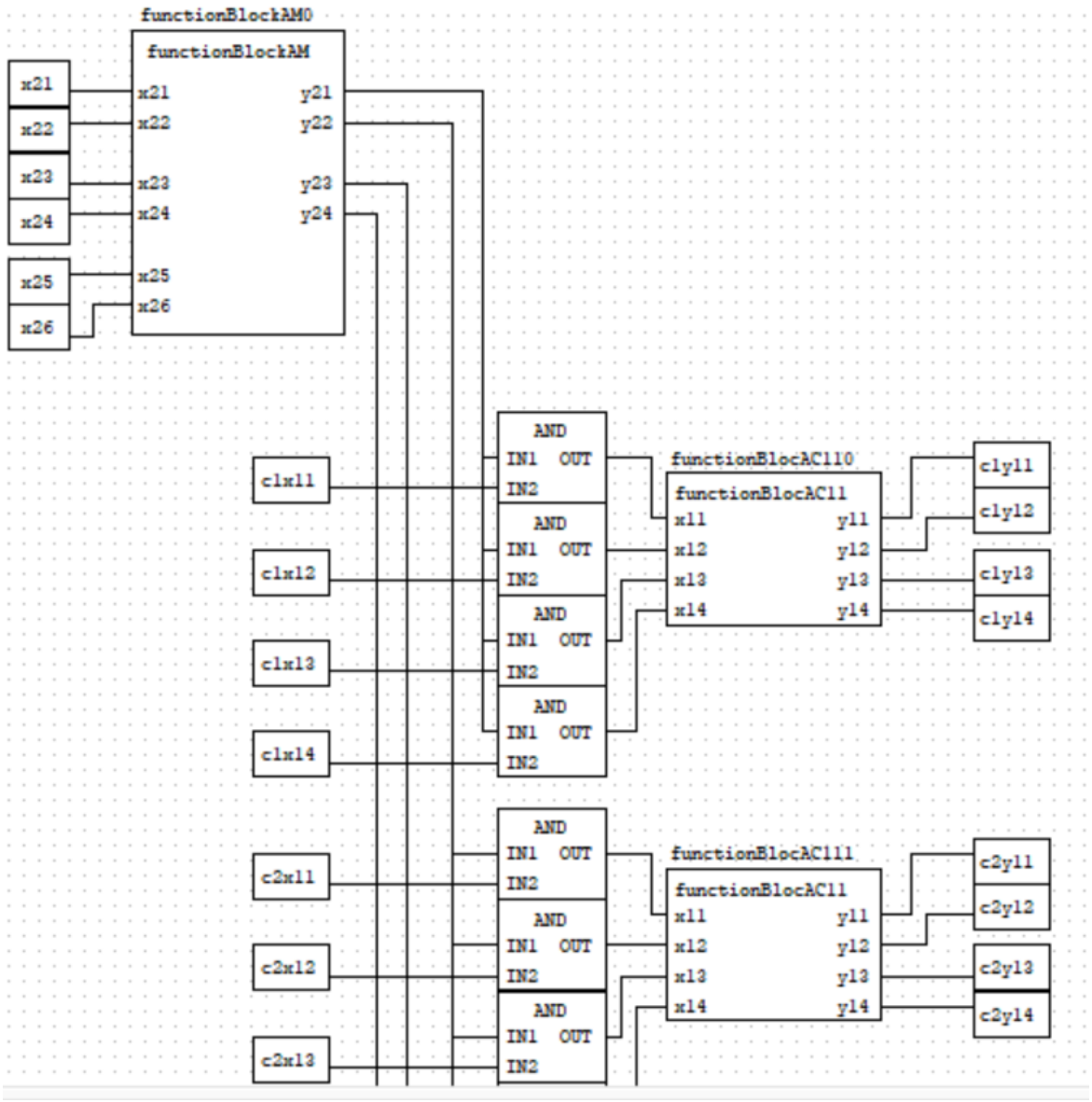


Рисунок В.6 – FBD представлення програми TEIC частина 1

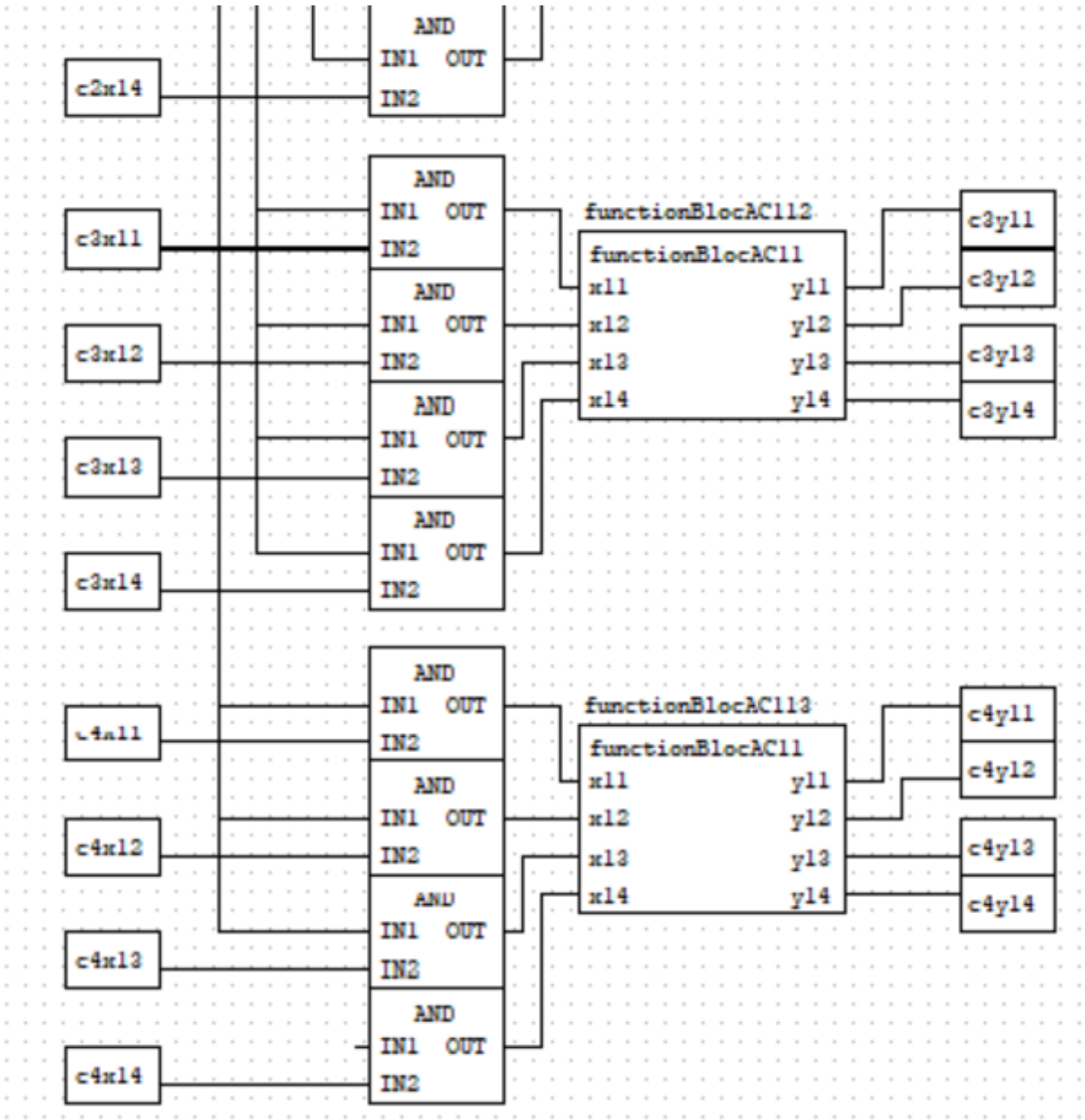


Рисунок В.7 – FBD представлення програми TEIC частина 2

ДОДАТОК Г

Диплом переможця всеукраїнського конкурсу

Рисунок Г.1 – Диплом

ДОДАТОК Д.

Перелік праць студента за темою роботи

1. Modeling and Programming of Elements of Integrated Systems in Industrial Controller Languages [Text] / [M. Poliakov, B. Zhurakovskyi,. O. Poliakov et. al.] // CEUR Workshop Proceedings 3702, 2024. – P. 169-178.

2. Poliakov, M. Interoperability of Integrated Hierarchial Systems [Text] / M. O. Poliakov, S. O. Subbotin, O. M. Poliakov // Системні технології : регіональний міжвузівський збірник наукових праць. – 2021. – Вип. 2 (133). – С. 68–78.

3. Poliakov, M. Prospects for studying cognitive control systems in technical universities [Electronic resource] / M. Poliakov, O. Stepanenko, O. Poliakov / International Conference on Academic Studies in Technology and Education (ICASTE), November 12-15, 2025, Antalya, Turkiye : Abstract Book – P. 9. – Access mode :
https://www.arste.org/storage/ICASTE/2025/ICASTE2025_Abstract.pdf

4. Poliakov, M. Structure Of he Course On Studying Functional Extensions of Fsm for Design Hierarchical Integrated Control Systems [Electronic resource] / [M. Poliakov, A. Pirozhok, O. Poliakov et. al.] // International Conference on Studies in Engineering, Science, and Technology (ICSEST) 06 - 10 November 2025, Istanbul, Turkey. – Abstract Book – P. 45.

5. Поляков, О. М. Прототипування пристроїв керування систем з промисловими контролерами [Текст] / О. М. Поляков, Б.Ю. Жураковський // Системні технології. – Дніпро. – 2024. – 2(151). – С. 50-61. – DOI 10.34185/1562-9945-2-151-202 4-05

6. Поляков, О. М. Методика прототипування пристроїв керування на базі додатку OpenPLC [Текст] / О. М. Поляков, Б. Ю. Жураковський] // In. International scientific and technical conference Information Technologies in Metallurgy and Machine building – ITMM 2024. – Dnipro. – 10-11, April. – 2024. – P. 376-382. – DOI: 10.34185/1991-7848.itmm.2024.01.072.

7. Poliakov, M. O. Principles of construction of the information and control system of the department of the university [Text] / [M. O. Poliakov, P. D.

Andrienko, O. M. Poliakov et al.] // System technologies). – Dnipro. – 2023. – N 2(145). – P.30 – 42. – DOI 10.34185/1562-9945-2-145-2023-04.

8. Poliakov, M. Set-theoretical FSM models activity subsystem for Cognitive Control Systems. / M. Poliakov, S. Subbotin, O. Poliakov // In Proceeding of the 15th International Conference ”The Experience of Designing and Application of CAD Systems” (CADSM), (26 February - 2 March, 2019, Polyana-Svalyava (Zakarpattya), Ukraine). – P. 1/1 – 1/4

ДОДАТОК Е
Слайди презентації

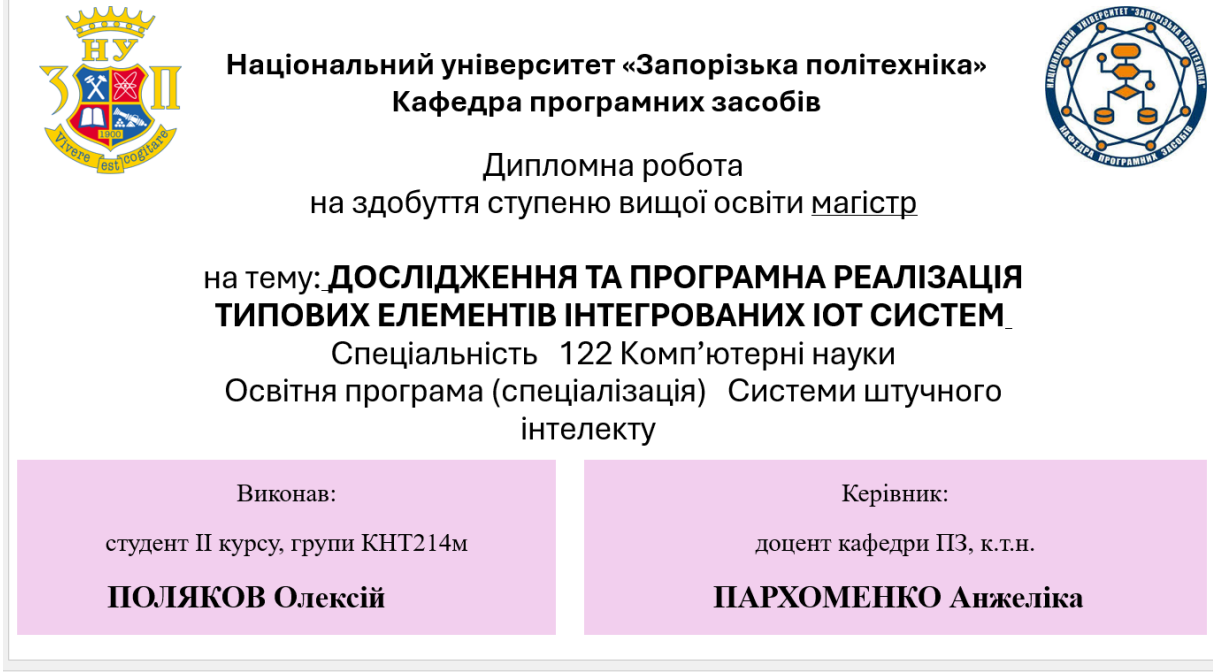


Рисунок Е.1 – Слайд 1

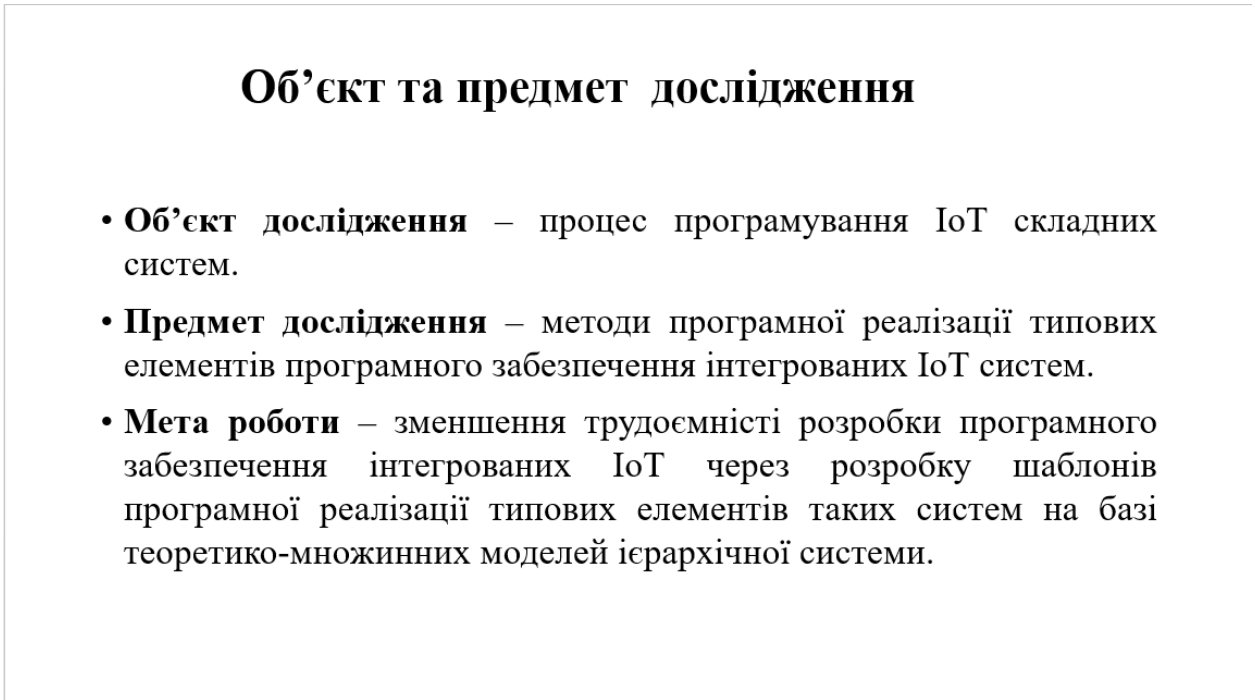
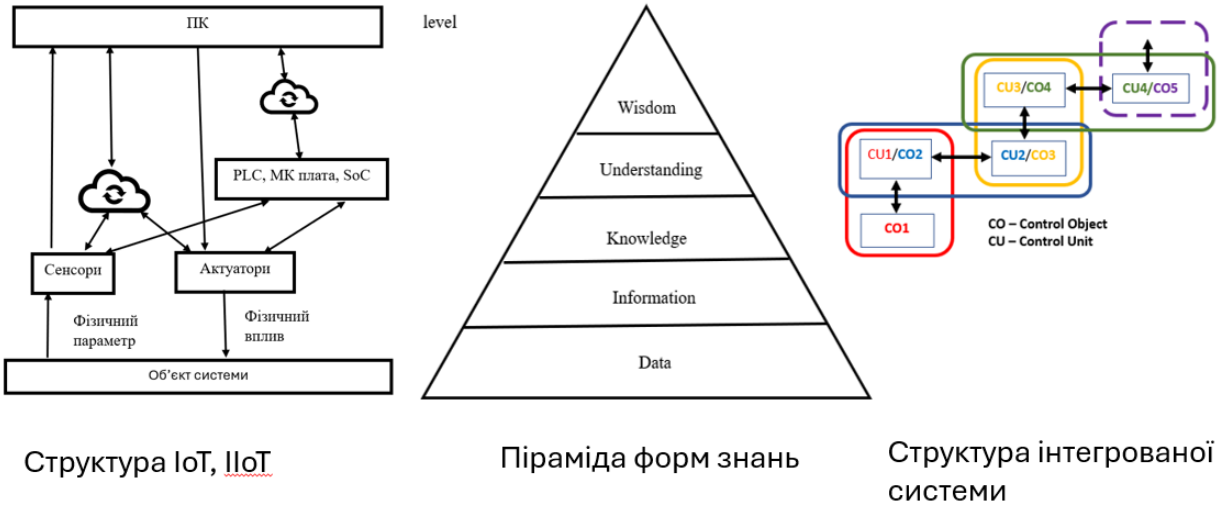


Рисунок Е.2 – Слайд 2

Аналіз предметної області



Структура IoT, IIoT

Піраміда форм знань

Структура інтегрованої системи

Рисунок Е.3– Слайд 3

Аналіз мов програмування

Позначення	Найменування	Тип мови	Область ефективного застосування
LD	Ladder Diagram, драбинні діаграми	Графічна	Логіка взаємного блокування
SFC	Sequential Function Chart, послідовні функціональні кроки	Графічна	Функціональна специфікація системи, керування послідовністю кроків
FBD	Function Block Diagram, функціональні блокові діаграми	Графічна	Процеси безперервного керування, складні обчислювання
ST	Structured Text, структурований текст	Текстова	Циклічні та розгалужені алгоритми, робота зі складними типами даних
IL	Instruction List, список інструкцій	Текстова	Там де потрібна мінімізація часу виконання та обсягу пам`яті

Рисунок Е.4 – Слайд 4

Вибір середовища програмування

Характеристика	Arduino IDE	RSLogix	OpenPLC	FPLOG
1. Вільне розповсюдження	+	-	+	+
2. Незалежність від типу PLC	-	-	+	-
3. Підтримка усіх мов стандарту IEC 61131-3	-	-	+	-
4. Вбудований симулятор контролера	-	-	+	+
5. Низька вартість	+	-	+	+
6. Сумісність зі платформою Arduino	+	-	+	-

Рисунок Е.5 – Слайд 5

Програмна модель керуючого автомату мовою С

<S, X, Y, s0, δ, λ>,

де S – скінченна непорожня множина станів; X – скінченна непорожня множина входів (вхідний алфавіт); Y – скінченна непорожня множина виходів (вихідний алфавіт); s0 – початковий стан; δ – функція переходу; λ – вихідна функція.

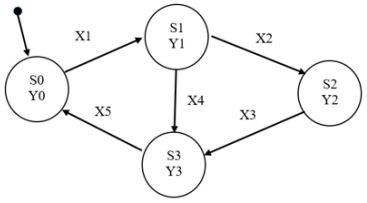
Поточний стан	Новий стан			
	S0	S1	S2	S3
S0		X1		
S1			X2	X4
S2				X3
S3	X1			

а

Стан	Вихід
S0	Y0
S1	Y1
S2	Y2
S3	Y3

б

Табличне завдання керуючого автомату



Приклад графу керуючого автомату

Вхідними данними для побудови програмної моделі скінченого автомата є його граф або таблиці переходів та виходів, початковий стан автомата. Елементам множин теоретико-множинного кортежу автомата відповідають змінні процедури автомату. усі змінні відносяться до бінарного типу даних.

Програмна модель скінченого автомата містить блоки задання початкового стану, виконання дій (виходів) у поточному стані, виконання переходу у новий стан. Процедура автомата виконується циклічно

Рисунок Е.6 – Слайд 6

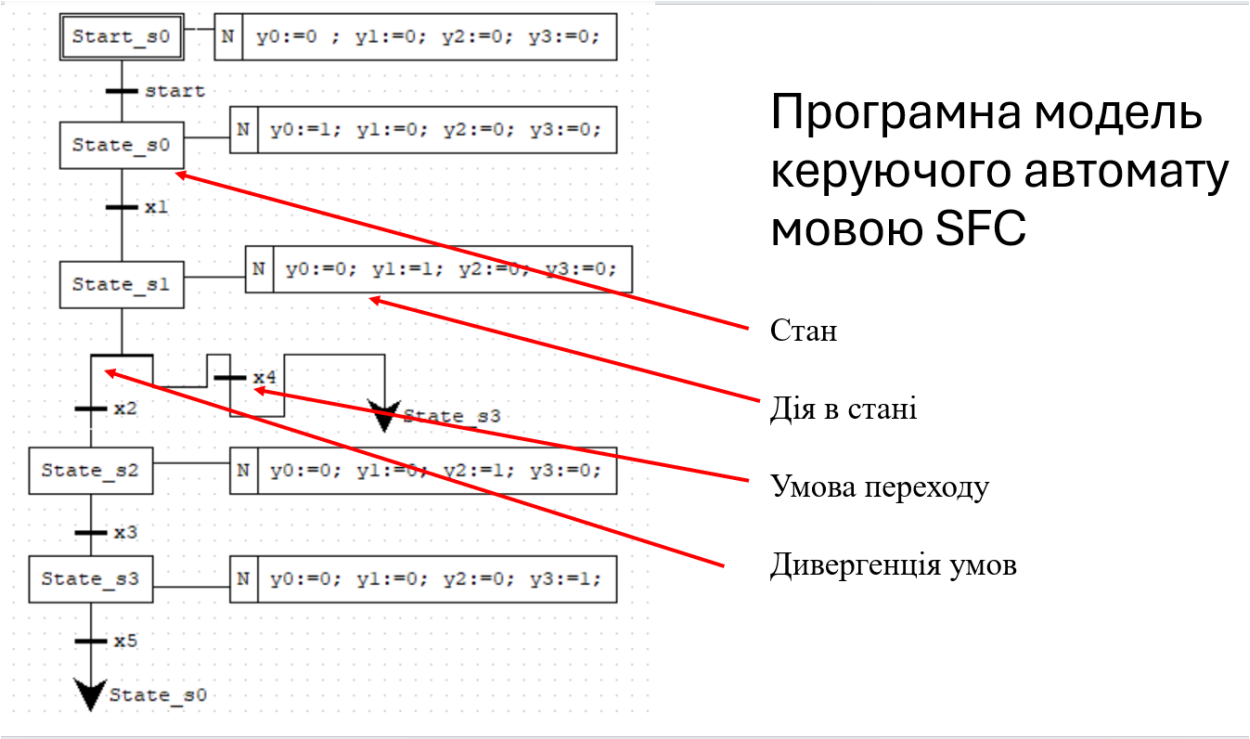


Рисунок Е.7 – Слайд 7

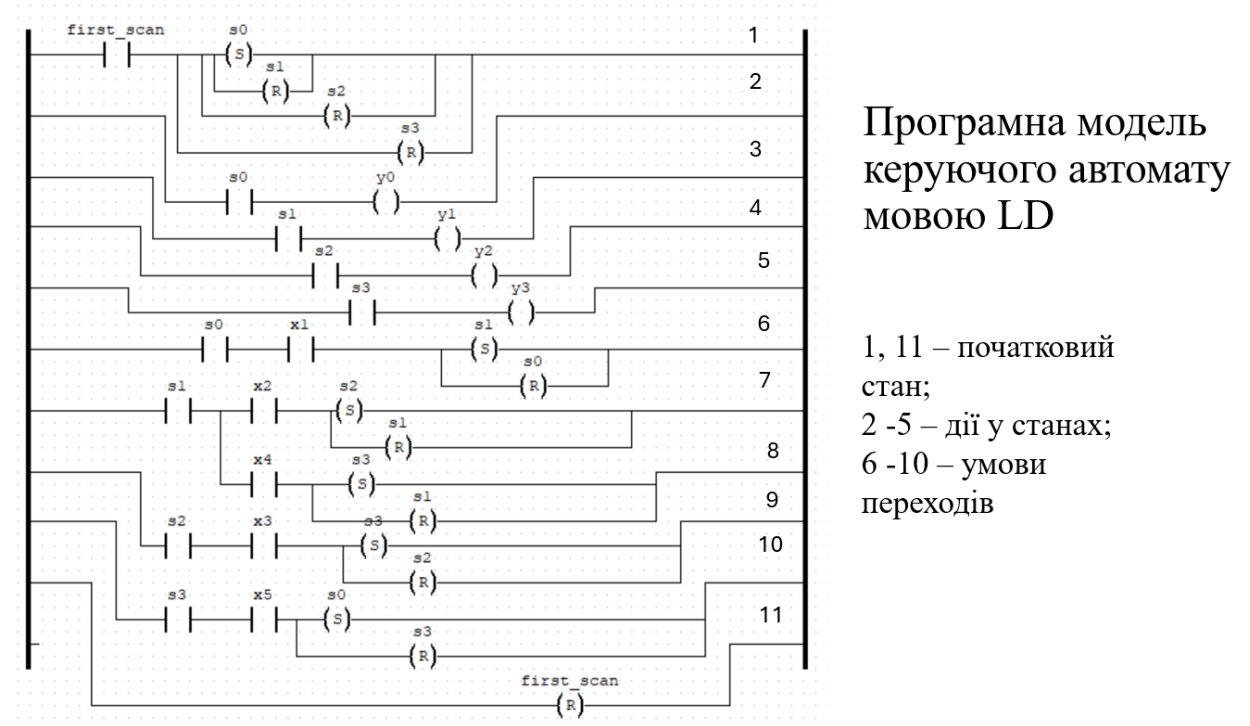
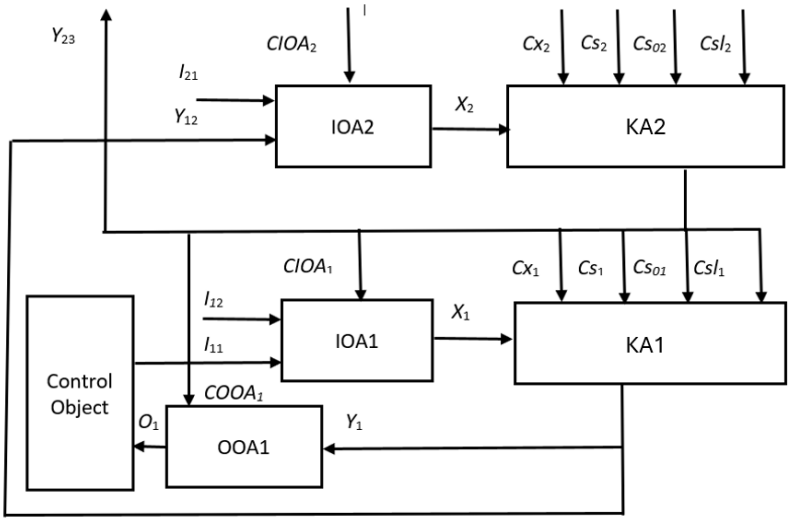


Рисунок Е.8 – Слайд 8

Структурна схема типового елемента інтегрованої системи



KA2-керуючий автомат рівня 2
KA1-керований автомат рівня 1
OOA1- вихідний операційний автомат рівня 1
IOA1-вхідний операційний автомат рівня 1
IOA2-вхідний операційний автомат рівня 2

Рисунок Е.9 – Слайд 9

Методика прототипування TEIC

Під прототипом системи на основі PLC розумітимемо її функціональний аналог, в якому PLC заміщений спрощеною мікроконтролерною платою. При цьому вихідний код керуючої програми, яка виконується в мікроконтролері плати, написаний мовами програмування за стандартом IEC 61131-3 та функціонально повторює програмний код системи оригіналу.

На першому етапі проводиться вибір мікроконтролерної плати з урахуванням необхідної кількості цифрових та аналогових каналів введення/виводу, каналів з ШІМ, оцінюється потреба у додаткових вузлах, які відсутні у платі прототипу

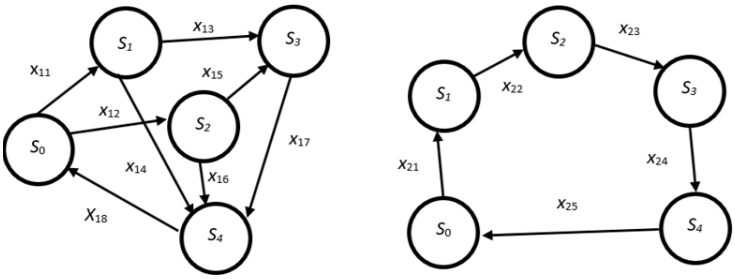
На другому етапі проводиться вибір структури та елементів POU для проекту прототипу керуючої програми. Програма є типом POU на верхньому рівні організації проекту прототипу.. Операційні автомати реалізуються як функції чи функціональні блоки, а управляючі – як функціональні блоки

На третьому етапі обираються мови програмування для POU проекту прототипу програми, що управляє.

Рисунок Е.10 – Слайд 10

Тестування елемента інтегрованої системи

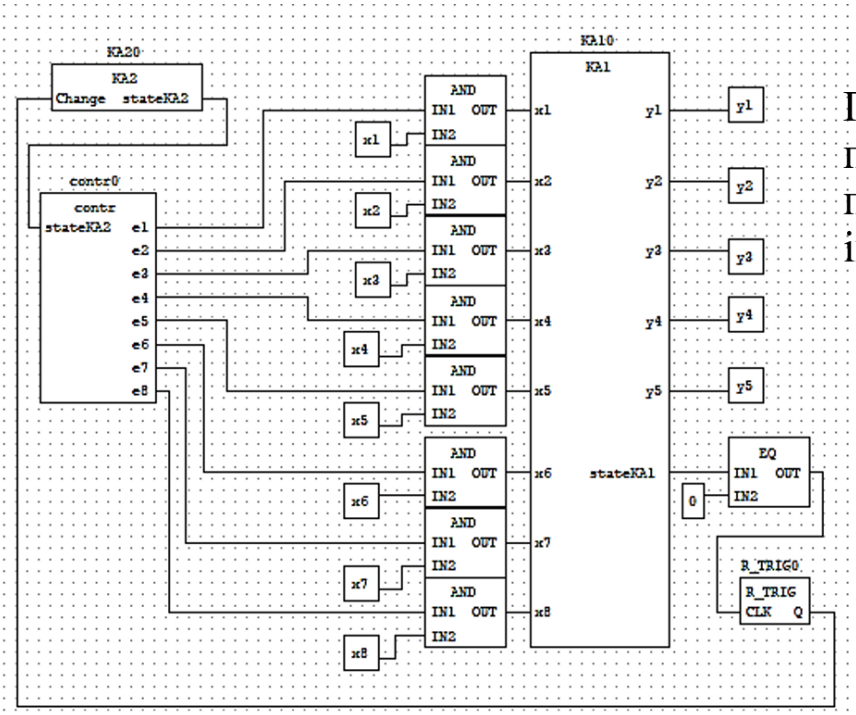
Граф керованого КА1 і керуючого КА2 автомата



Через вершину s0 в автоматі КА1 проходять шляхи чотирьох циклів управління об'єктом системи. Це шлях 1: s0-s1-s3-s4-s0; шлях 2: s0-s1-s4-s0; шлях 3: s0-s2-s3-s4-s0 і шлях 4: s0-s2-s4-s0. Цикл управління змінюється під час переходу активності через вершину s0.

Стратегія управління у тому, щоб активність автомата переміщалися в послідовності: шлях 1 – шлях 2 – шлях 3 – шлях 4. Далі послідовність повторюється знову. Стратегію управління формує автомат КА2, в якому є тільки один шлях і зміна стану якого відбувається в момент завершення чергового циклу автомата КА1.

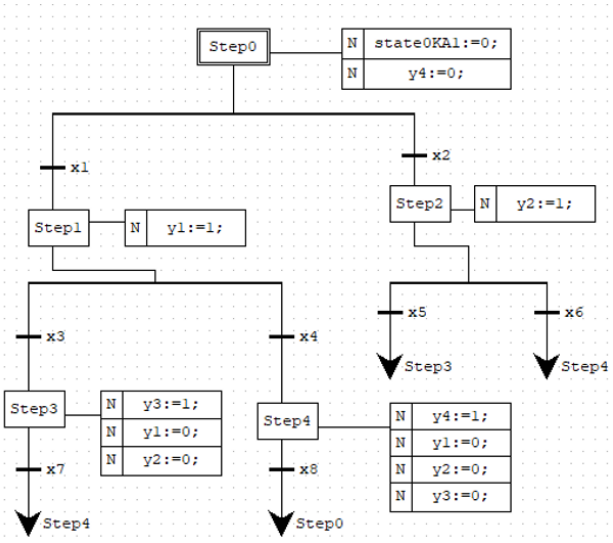
Рисунок Е.11 – Слайд 11



Графічне FBD представлення програми елемента інтегрованої системи

Рисунок Е.12 – Слайд 12

SFC функціональний блок автоматів КА1 та КА2



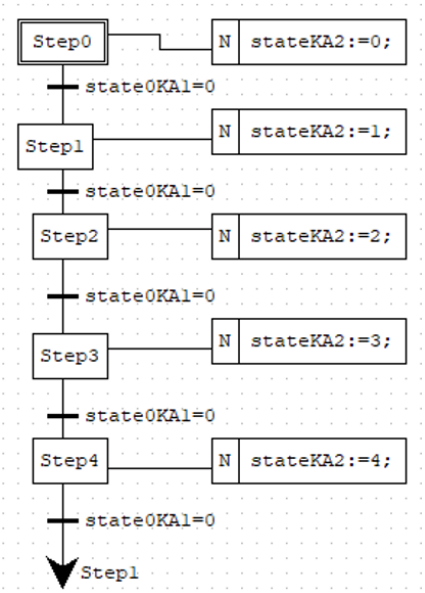
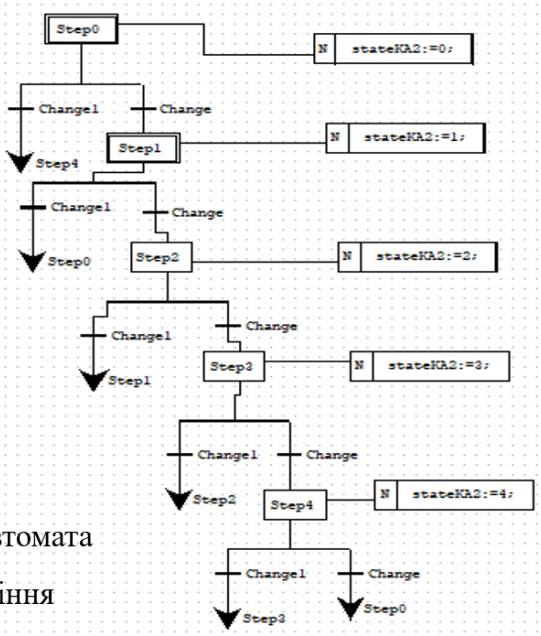


Рисунок Е.13 – Слайд 13

Елементи POU типового елемента

ST функціональний блок *contr* керування дозволами входів автомата КА1

```
CASE stateKA2 OF
  0: e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
  1: e1:=1; e2:=0; e3:=1; e4:=0; e5:=0; e6:=0; e7:=1; e8:=1;
  2: e1:=1; e2:=0; e3:=0; e4:=1; e5:=0; e6:=0; e7:=0; e8:=1;
  3: e1:=0; e2:=1; e3:=0; e4:=0; e5:=1; e6:=0; e7:=1; e8:=1;
  4: e1:=0; e2:=1; e3:=0; e4:=0; e5:=0; e6:=1; e7:=0; e8:=1;
  ELSE
    e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
    END_CASE;
```



SFC модель керованого автомата КА2 із двома стратегіями управління

Рисунок Е.14 – Слайд 14

Висновки

1. Проведений аналіз структури складних IoT систем висвітлив переваги побудови таких систем у вигляді ієрархічної інтегрованої структури з використанням формалізму керуючих автоматів та широкими можливостями для системної адаптації. Констатоване також відсутність програмної реалізації типових елементів інтегрованої системи, наявність у її складі PLC
2. Для програмної реалізації таких елементів обрані мова програмування C та мови за стандартом IEC 61131-3 а також середовище програмування OpenPLC. Запропонована методика прототипування дозволяє трансформувати операційні та керуючі автоматів проектних моделей системи керування у компоненти організації програм проекту прототипу системи.
3. Розроблена та протестована методика програмування керуючих автоматів мовами C та мовами стандарту IEC 61131-3. Розроблена структура типового елемента інтегрованої системи, який складається з комплексу керуючого, керованого та операційних автоматів, що забезпечує адаптивну поведінку системи щодо завдань її функціонування.

Рисунок Е.15 – Слайд 15

Висновки

5. Визначені типи програмної організації елементів типового елемента інтегрованої системи. Мовами PLC розроблені програми цих елементів. Програми протестовані з використанням середовища програмування OpenPLC та мікроконтролерних плат Arduino. Визначено, що застосування типового елемента інтегрованої системи на 10-15 відсотків зменшує трудоємність програмування завдань адаптивної поведінки складної IoT системи
6. Запропонована методика прототипування пристроїв на основі PLC може бути корисною у навчальному процесі підготовки магістрів
7. За темою магістратської роботи опублікована 7 наукових робіт.
8. Деякі положення роботи доповідались на Всеукраїнському конкурсі студентських наукових робіт «Інформаційні технології», де робота отримала Диплом III ступеня.

Рисунок Е.16 – Слайд 16