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)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ МОДЕЛЕЙ І СПОСОБІВ ОПРАЦЮВАННЯ ДАНИХ ПРИ ВЗАЄМОДІЇ З ВІДДАЛЕНОЮ ЛАБОРАТОРІЄЮ
DEVELOPMENT AND RESEARCH OF MODELS AND METHODS FOR DATA PROCESSING FOR REMOTE LABORATORY INTERACTION

Виконав: студент 4 курсу, групи КНТ-212
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

ПАРХОМЕНКО І.А.
(ПРІЗВИЩЕ та ініціали)
Керівник ТАБУНЩИК Г.В.
(ПРІЗВИЩЕ та ініціали)
Рецензент ШИТІКОВА О.В.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

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

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПАРХОМЕНКА Іллі Андрійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення та дослідження моделей і способів опрацювання даних при взаємодії з віддаленою лабораторією. Development and Research of Models and Methods for Data Processing for Remote Laboratory Interaction
керівник проєкту (роботи) к.т.н., проф., ТАБУНЩИК Галина Володимирівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від "07" квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 1 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження моделей і способів опрацювання даних при взаємодії з віддаленою лабораторією. 3. Проєктування програмно-апаратного забезпечення системи. 4. Розробка та практичне використання програмного забезпечення для опрацювання даних у віддаленій лабораторії.

5. Перелік графічного матеріалу (з 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	ТАБУНЩИК Г.В., професор		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз предметної області.	2 тиждень	Розділ 1
3	Дослідження способів подання та опрацювання даних у віддаленій лабораторії	3 тиждень	Розділ 2
4	Розробка архітектури програми. Вибір мови програмування та технологій розробки	4 тиждень	Розділ 3
5	Розробка програми.	4-5 тиждень	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6-7 тиждень	ПЗ, Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент _____ Ілля ПАРХОМЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Галина ТАБУНЩИК
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 105 с., 13 табл., 37 рис., 2 дод., 51 джерело.

ВІДДАЛЕНА ЛАБОРАТОРІЯ, ОПРАЦЮВАННЯ ДАНИХ, FSM МОДЕЛЬ, HYBRID TAKE-HOME-LAB, МІКРОКОНТРОЛЕР, ПРОГРАМНО-АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт дослідження – процес опрацювання даних в інфраструктурі віддаленої лабораторії GOLDi 2.0.

Предмет дослідження – моделі та способи опрацювання даних при взаємодії з віддаленою лабораторією.

Мета роботи – покращення користувацького досвіду при взаємодії з віддаленою лабораторією за рахунок моделей та способів інтегрованого опрацювання даних.

Матеріали, методи та технічні засоби: структурне та об’єктно-орієнтоване програмування, мова програмування C++, фреймворк Arduino Core, кейс Hybrid Take-Home-Lab з платою Arduino, мікроконтролер ATmega2560, середовище програмування Arduino IDE, персональний комп’ютер з процесором Intel Core i7-13700HX під управлінням операційної системи Microsoft Windows 11 Home.

Результати. Досліджено та розроблено моделі, способи та програмне забезпечення для інтегрованого опрацювання даних віддаленого експерименту в інфраструктурі лабораторії GOLDi 2.0.

Висновки. Розроблені моделі, способи та програмне забезпечення для інтегрованого опрацювання даних віддаленого експерименту покращують взаємодію користувачів з середовищем GOLDi 2.0, що забезпечує набуття студентами відповідних знань та практичних навичок.

Галузь використання – навчальний процес в закладах вищої освіти для студентів галузі знань 12 «Інформаційні технології».

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 105 pages, 13 tables, 37 figures, 2 appendixes, 51 sources.

REMOTE LABORATORY, DATA PROCESSING, FSM MODEL, HYBRID TAKE-HOME-LAB, MICROCONTROLLER, SOFTWARE AND HARDWARE.

Object of the study is the process of data processing in the GOLDi 2.0 remote laboratory infrastructure.

Subject of the study is models and methods of data processing during the interaction with a remote laboratory.

The purpose of the work is to improve the user experience when interacting with a remote laboratory through models and methods of integrated data processing.

Materials, methods and technical tools: structural and object-oriented programming, C++ programming language, Arduino Core framework, Hybrid Take-Home-Lab case with an Arduino board, ATmega2560 microcontroller, Arduino IDE programming environment, personal computer with an Intel Core i7-13700HX processor running the Microsoft Windows 11 Home operating system.

Results. Models, methods and software for integrated processing of remote experiment data in the GOLDi 2.0 laboratory infrastructure were researched and developed.

Conclusions. The developed models, methods and software for integrated processing of remote experiment data improve user interaction with the GOLDi 2.0 environment, which ensures that students acquire relevant knowledge and practical skills.

The field of application is an educational process in higher education institutions for students in the field of study 12 "Information Technologies".

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;
AR – Augmented Reality;
DaaS – Device as a Service;
ECP – Experiment Control Panel;
FPGA – Field-Programmable Gate Array;
FSM – Finite State Machine;
GPIO – General Purpose Input/Output;
GUI – Grafical User Interface;
I2C – Inter-Integrated Circuit;
IDE – Integrated Development Environment;
IoT – Internet of Things;
LaaS – Lab as a Service;
LMS – Learning Management System;
LTI – Learning Tools Interoperability;
PCB – Printed Circuit Board;
PLD – Programmable Logic Device;
PWM – Pulse-Width Modulation;
RTOS – Real-Time Operating System;
SOA – Service-Oriented Architecture;
SPI – Serial Peripheral Interface;
SSO – Single Sign-On;
VHDL – VHSIC Hardware Description Language;
VHSIC – Very High Speed Integrated Circuit;
VR – Virtual Reality;
WIDE – Web IDE;
ВЛ – віддалена лабораторія;
МК – мікроконтролер;
ОЗП – оперативний запам'ятовуючий пристрій;
ПЗ – програмне забезпечення;
ПК – персональний комп’ютер.

ВСТУП

На сьогоднішній день актуальність подальшого розвитку цифрових лабораторій (віддалених, віртуальної/доповненої реальності, симуляторів тощо) підтверджена поточним станом в цій галузі, а також перевагами їх використання для дистанційного навчання як в умовах пандемії Covid-19 [1]-[2], так і в умовах війни в Україні.

Хоча потенціал цих лабораторій очевидний, їх реалізації є специфічними, а окремі дидактичні цілі не є взаємозамінними між різними країнами, університетами та дисциплінами. Їх використання складно перенести в контекст (попередні знання студентів, індивідуальний фокус, обсяг завдання) іншого університету, а отже їм бракує сумісності. Крім того доступність та гнучкість для студентів залишається проблематичною, тож зазвичай вони локально використовуються в межах окремого університету та доступні лише обмеженій групі студентів [3].

Саме тому актуальною задачею є подальший розвиток дидактичних, технічних та організаційних рішень, що дозволять створити відкрите цифрове міжуніверситетське навчальне середовище, яке можна адаптувати до вимог навчання та потреб студентів, що надасть їм навички, необхідні для майбутньої професійної діяльності.

Проєкт Hybrid Take-Home-Lab спрямований на підтримку компетентнісного викладання та навчання шляхом проведення складних віртуальних та віддалених лабораторних експериментів студентом з дому, використовуючи власні доступні ресурси, які комбінуються за потреби для навчання, орієнтованого на студента.

Мета роботи – покращення користувацького досвіду при взаємодії з віддаленою лабораторією за рахунок моделей та способів інтегрованого опрацювання даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд проблематики в галузі віддалених лабораторій

У всьому світі лабораторна складова навчальних курсів є невід'ємною частиною різних програм навчання на всіх рівнях підготовки, а лабораторії мають вирішальне значення в освіті студентів [4]-[5]. Під час виконання лабораторних робіт студенти отримують глибше розуміння теоретичних знань завдяки практичному досвіду та отримують чітке уявлення про робоче середовище в обраній ними професії [6]. Однак забезпечення необхідної інфраструктури є величезним викликом з точки зору високих витрат на придбання та обслуговування лабораторного обладнання.

Сьогодні у сфері створення та використання університетських навчальних лабораторій спостерігаються такі тенденції:

– впровадження компетентнісних підходів до студентоорієнтованого навчання [7], а також навчання, орієнтованого на проблеми та дослідження [8];

– цифровізація навчально-методичних середовищ шляхом створення лабораторій крос-реальності на основі використання методів та засобів віртуальної/доповненої реальності (VR/AR) та віддаленої інженерії [8].

Остання тенденція розширює спектр онлайн-лабораторій навчання та спрямована на створення змінного дидактичного навчально-методичного середовища для орієнтованого на цільову аудиторію представлення тематично релевантного контенту як для студентів, так і для викладачів [3].

Використання цифрових лабораторій дозволяє викладачам розвивати креативність та інноваційні навички студентів [9]. Крім того, фундаментальні елементи Індустрії 4.0, а саме інформаційна прозорість, мережева взаємодія та децентралізоване прийняття рішень, можуть бути інтегровані в існуючі або нові експерименти за допомогою лабораторій крос-реальності [10].

Віддалений доступ до навчальних ресурсів відкриває гнучкий графік, який відповідає індивідуальному робочому ритму та рівню знань кожного студента.

Але, наприклад, хоча успішне впровадження гібридної лабораторії GOLDi [11]-[12] не викликає сумнівів, воно передбачає використання індивідуально виготовленого обладнання, вартість якого є достатньо високою, а обслуговування складним.

Тому подальший розвиток технічних, дидактичних та організаційних рішень для відкритих цифрових лабораторій, що дозволить викладачам розробляти орієнтовані на потреби, студентоцентровані та вільно комбіновані навчальні матеріали є актуальним та передбачає:

– міжтипове поєднання різних типів лабораторій;
– міжелементний склад окремих лабораторних об'єктів;
– міждисциплінарне використання цифрових лабораторій;
– міжуніверситетський доступ до цифрових репозиторіїв [3].

1.2 Аналіз особливостей інфраструктур віддалених лабораторій

Як показали проведені дослідження, онлайн лабораторії пройшли декілька стадій розвитку. Ранні рішення були зосереджені на фактичному застосуванні індивідуально розробленої вебінфраструктури, але останні два десятиліття спостерігалася тенденція до проєктування універсальної архітектури інфраструктури, яку можна використовувати незалежно від фактичної реалізації [13]-[14].

Специфікація універсальних фреймворків та інтерфейсів заклала основу для третього покоління віддалених лабораторій (ВЛ) – спільних лабораторій [15]. Тут ідея постійної доступності була продовжена та перенесена на міжуніверситетські лабораторії. Отже, студенти університетів-партнерів могли використовувати інфраструктуру один одного, що вимагало розширень фреймворків, особливо на адміністративному та організаційному

рівнях, для забезпечення захисту даних, координації доступу та оцінки результатів.

Але усі попередні реалізації не досягли бажаної гнучкості цифрових лабораторій, оскільки конкретний навчальний контент зазвичай визначався один раз на етапі проєктування, а коригування з боку конфігурації лабораторного об'єкта, а також інтерфейсу користувача, були можливими лише за умови глибокого розуміння вебтехнологій на рівні програмування. Тому потрібно було усунути цю прогалину та створити основу для більш інтенсивного використання орієнтованих на навчання налаштовуваних цифрових лабораторій для міжуніверситетського використання [3].

Один з підходів, який дозволяє точніше адаптувати експерименти до конкретних навчальних цілей, базується на розкладанні експерименту на окремі елементи, що створює можливість їх заміни на основі підходу hardware/software in-the-loop [15].

Крім того, лабораторні модулі також можна розподілити по різних локаціях. Наприклад, датчики та виконавчі механізми доступні в одному університеті, а інший бере на себе завдання забезпечення різних архітектур контролерів, які потім також можна включити до зовсім інших експериментів. Водночас цифрові двійники віддалених експериментів можуть бути встановлені в різних місцях, що забезпечить покращений захист від збоїв [3].

1.2.1 Лабораторія віддаленого мікроконтролера

Дозволяє дослідити можливості різних фреймворків ВЛ, а також способи інтеграції мікролабораторій в ідею більшої лабораторної інфраструктури. Її основою є платформа дистанційного навчання з відкритим кодом Edrys, яка сприяє розробці та спільному використанню пристроїв дистанційного навчання та дозволяє ділитися локальною конфігурацією лише за допомогою веббраузера [3].

Центральний сервер використовується як відправна точка та точка зустрічі, але зареєстрований учасник може ділитися своєю локальною конфігурацією/обладнанням, просто відкривши спеціальний вебсайт Edrys, який працює в режимі «станції». Цей вебсайт працює як ретранслятор між локальним обладнанням, до якого має бути доступ і яким має керувати локальний сервер, та віддаленим сервером Edrys. Система публікації/ підписки використовується для надсилання/дзеркалювання всіх повідомлень між станцією та всіма студентами, які працюють на цій конкретній станції, як показано на рис. 1.1 [3].

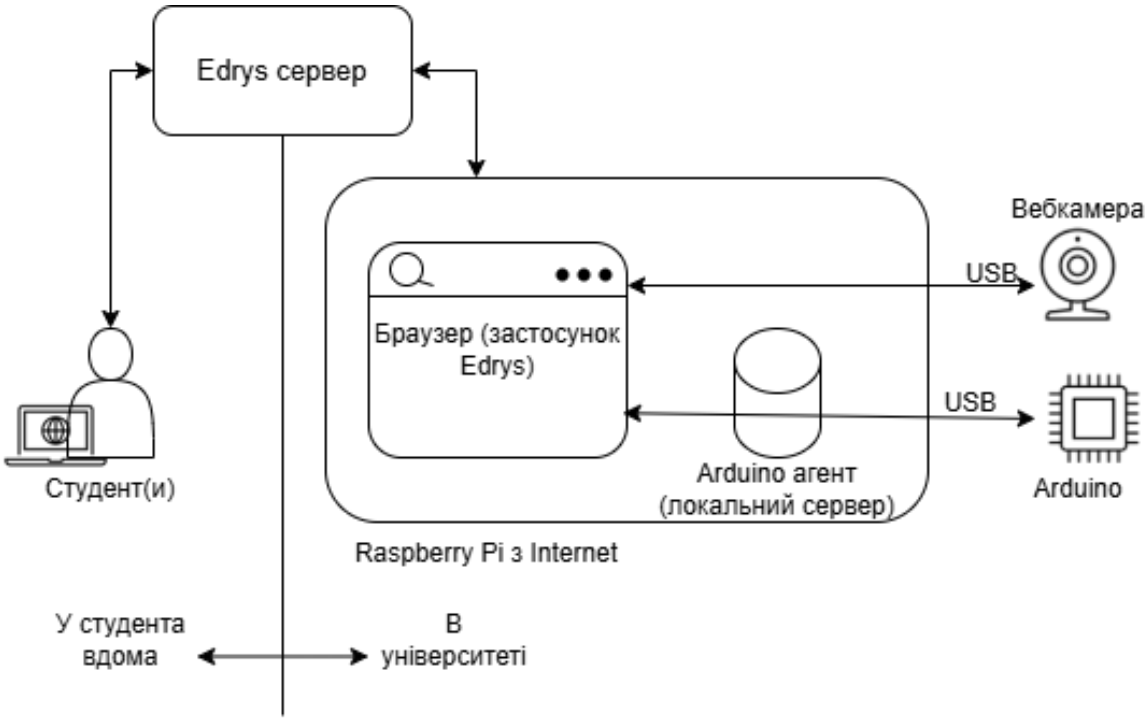


Рисунок 1.1 – Приклад інфраструктури Edrys [3]

Просте налаштування та розробка плагінів дозволяють створювати базові лабораторії, які складаються з редактора, терміналу та модуля вебкамери. Хоча ця конфігурація є статичною, можливо надавати різні форми дидактичного/освітнього контенту за допомогою модуля LiaScript для створення онлайн-курсів з розширеним синтаксисом Markdown [16]. Курси розглядаються як проєкти з відкритим кодом і можуть розміщуватися та публікуватися розподілено, без потреби в централізованому сервері. Крім

того, існують плагіни для LiaScript, які дозволяють програмувати симульовані платформи Arduino безпосередньо в браузері або вивчати програмування різними мовами.

Отже, користувачам не потрібно займати апаратне забезпечення станції, а натомість вони можуть проводити прості експерименти та навчання в цьому спрощеному середовищі. Тож один курс може містити різні симульовані лабораторні роботи та навчальні екземпляри програмування, курси можна створювати та адаптувати для студентів з різних дисциплін та з різним досвідом. До того ж LiaScript дозволяє автоматично перекладати свої курси різними мовами з точними налаштуваннями для захисту частин курсу від перекладу (наприклад, таких як код).

1.2.2 Гібридна лабораторія GOLDi 2.0

На відміну від ранньої концепції гібридної лабораторії GOLDi (GOLDi – Grid of Online Lab Devices Ilmenau [11], [17]-[18]), метою GOLDi 2.0 є розробка нового покоління ВЛ, в якій кілька лабораторних пристроїв можна буде об'єднати в систему віддаленого керування лабораторією. Це дозволить обмінюватися окремими пристроями або цілими експериментами між університетами (рис. 1.2).

У GOLDi 2.0 онлайн-лабораторія більше не розглядається як сукупність монолітно побудованих експериментів, а як набір лабораторних пристроїв, які взаємодіють один з одним, що відкриває абсолютно нові можливості для експериментів, а саме:

– можливість керувати апаратною моделлю в одному місці за допомогою блоку керування з іншого місця в межах одного експерименту;

– можливість під час експерименту дистанційно керувати лабораторними пристроями з вимогами жорсткого реального часу через Інтернет, підключивши лабораторний пристрій, відповідальний за критичну за часом частину, безпосередньо до моделі, тоді як керування, яке не є

критичним за часом, можна буде здійснювати через віддалений лабораторний елемент;

– можливість для великих електромеханічних апаратних моделей з величезною кількістю входів і виходів керування з багатьох різних місць.

Відтак, можлива паралелізація експериментів [3].

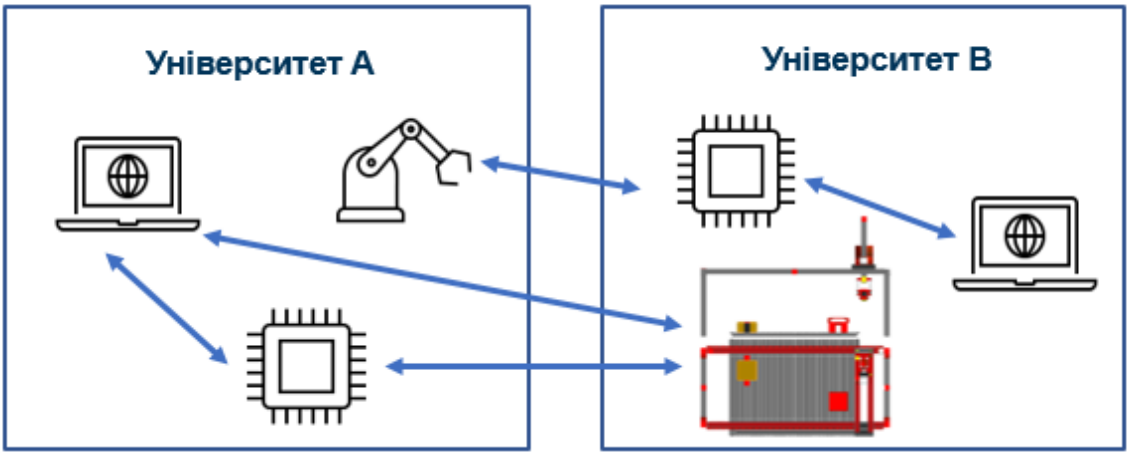


Рисунок 1.2 – Спільне використання лабораторного обладнання між університетами в GOLDi 2.0 [3]

Отже, GOLDi 2.0 дозволяє проводити симуляції, створювати віртуальні лабораторні середовища та віддалені лабораторні експерименти в єдиному навчальному середовищі, щоб ефективно відповідати вимогам навчання на основі компетенцій.

GOLDi 2.0 також підтримує ідею гібридної домашньої лабораторії для зацікавлених студентів, які мають вдома власне обладнання (Arduino, Raspberry Pi, демонстраційна плата Field-Programmable Gate Array (FPGA) або набори для експериментів з цифровою логікою). Тож GOLDi 2.0 надає студентам інтерфейсний блок, який вони можуть використовувати для легкого підключення з дому власних блоків керування до складних моделей обладнання в університеті, створюючи гібридну домашню лабораторію [3].

Hybrid Take-Home-Lab – це онлайн-лабораторія, яка пропонує експерименти як комбінацію мобільних лабораторних елементів, які можна

експлуатувати без доступу до лабораторії, з іншими лабораторними елементами, які можуть бути стаціонарними. Hybrid Take-Home-Lab розширює класичну класифікацію онлайн лабораторій ще одним виміром, як показано на рис. 1.3 [18].

Рисунок 1.3 – Місце Hybrid Take-Home-Lab в класифікації онлайн лабораторій [18]

ВЛ дозволяють проводити дистанційні експерименти через вебдоступ до реальних елементів керування та фізичних систем (електромеханічних моделей обладнання). Віртуальні лабораторії працюють виключно у віртуальних штучних світах. Гібридні лабораторії поєднують обидва підходи, дозволяючи, окрім дистанційно керованих експериментів за допомогою моделювання, працювати з віртуальними пристроями, які за своїми основними властивостями відповідають реальним пристроям. Цей підхід відповідає ідеї цифрових двійників, яка має велике значення для розвитку робочих середовищ Індустрії 4.0 [19].

Онлайн-лабораторія GOLDi гнучко реалізує всі варіанти. Це дає змогу проводити всі експерименти або повністю віртуально, або на реальних пристроях, або в комбінації обох. На основі гнучкої грід структури експеримент складається з двох компонентів. По-перше, є різні блоки керування (наприклад, мікроконтролер, FPGA та ін.). По-друге, є електромеханічні фізичні системи (наприклад, ліфт, 3-осьова модель або склад) (рис. 1.4). Кожну фізичну систему також можна моделювати у віртуальному середовищі. Взаємозв'язок між блоками керування та обраною моделлю фізичного обладнання під час віддаленого лабораторного сеансу роботи (експерименту), а також керування користувачами здійснюється лабораторним сервером, який також керує вебкамерами [19].

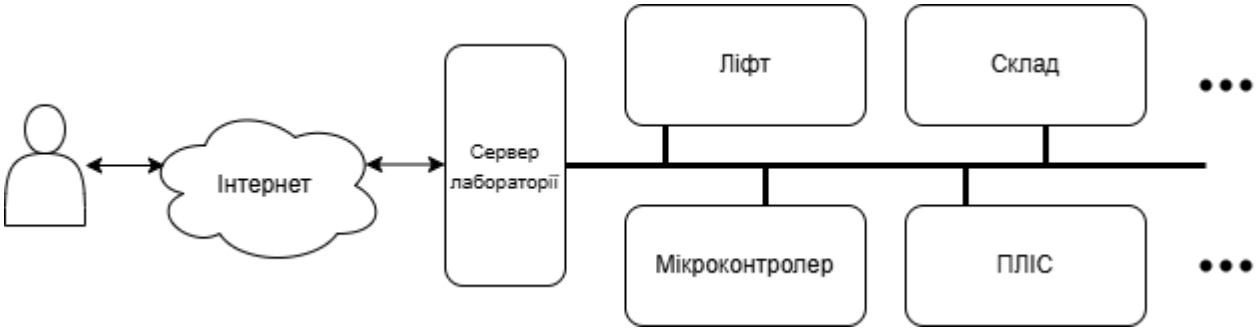


Рисунок 1.4 – Грід структура лабораторії GOLDi [19]

Hybrid Take-Home-Lab - це форма ВЛ, що складається з одного або кількох локальних пристроїв, підключених до віддаленого лабораторного обладнання (рис.1.5).

Наприклад, студент може запрограмувати контролер ліфта на мікроконтролері (МК) вдома. Цей МК буде підключений лише до інтерфейсного блоку, який з'єднує його з моделлю ліфта (рис. 1.6) у гібридній лабораторії GOLDi 2.0 [18].

Отже, експеримент наближається до реального експерименту для студентів, а типове завдання програмування охоплює не лише створення бінарних файлів, але й те, як їх фізично перенести на МК, чого ВЛ не пропонували в минулому [18].

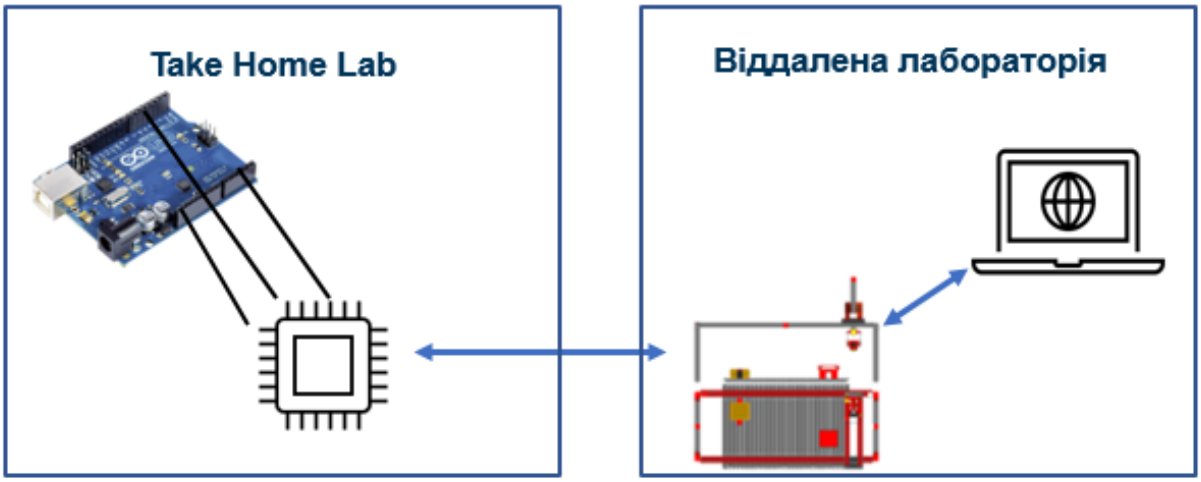


Рисунок 1.5 – Взаємодія Take-Home-Lab та гібридної лабораторії GOLDi 2.0 [3]

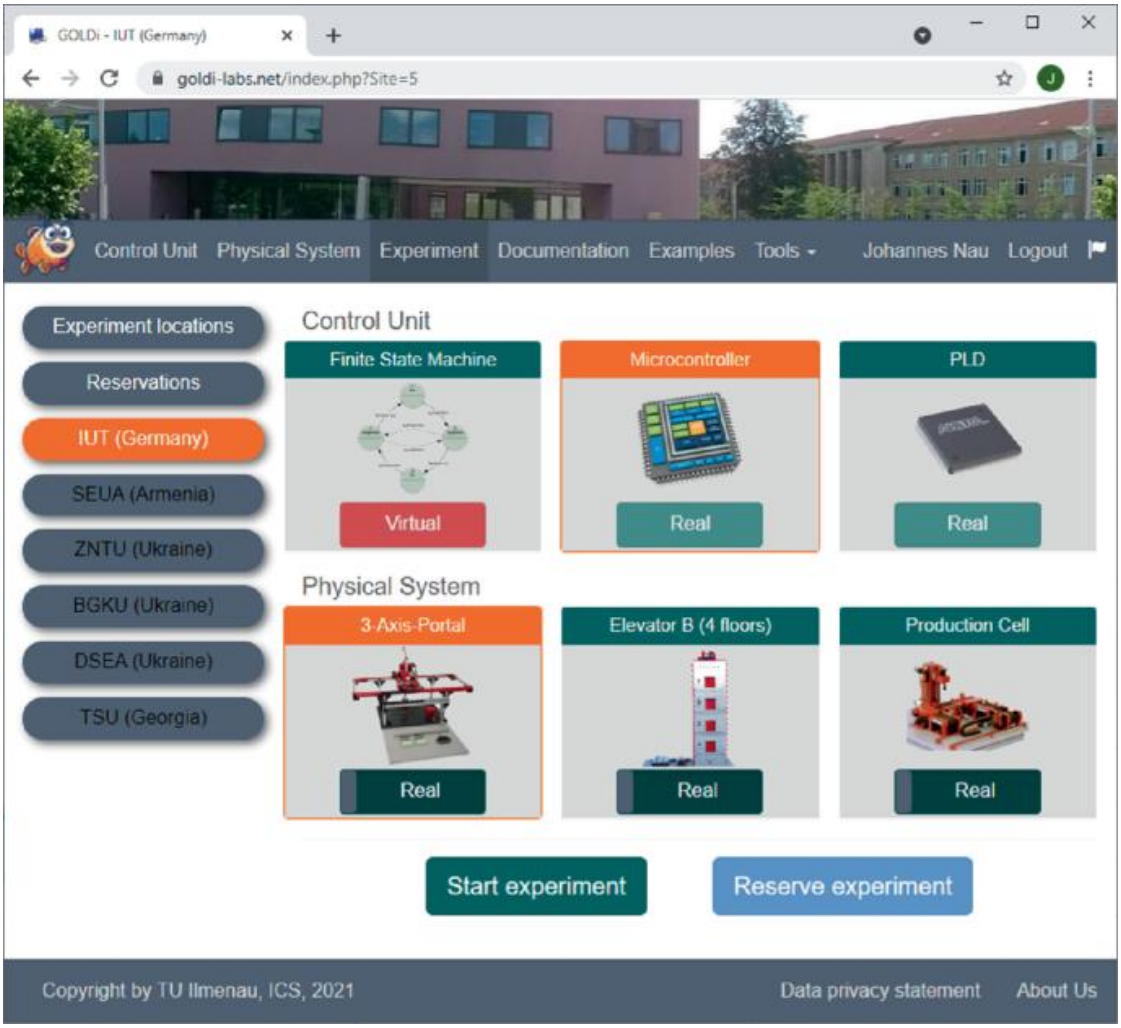


Рисунок 1.6 – Контрольні пристрої та фізичні системи в гібридній лабораторії GOLDi [18]

Наявним домашнім лабораторіям зазвичай бракує інтерфейсу з навчальною інфраструктурою та вони не пропонують можливості керувати реальними системами, які часто є занадто дорогими та занадто великими, щоб включати їх у домашню лабораторію.

Технічно це не сильно відрізняється від стандартної лабораторії з кількома дистанційно з'єднаними лабораторними пристроями, однак такий варіант використання необхідно враховувати з огляду на архітектурні особливості, оскільки він впливає на керування користувачами та пристроями [18].

1.2.3 Віддалена лабораторія VISIR

Дортмундський технічний університет (TUDO) має один із 18 екземплярів ВЛ VISIR (Virtual Instrumentation Systems in Reality) [20], яка дозволяє планувати та проводити віддалені експерименти з електроніки. За допомогою комп'ютерної миші або сенсорної панелі схеми можна віртуально створювати на графічному інтерфейсі користувача (GUI), який відтворює типову макетну плату для реалізації електронних схем (рис.1.7) [21].

З точки зору програмного забезпечення (ПЗ), VISIR поділяється на такі компоненти:

– клієнт VISIR (вебінтерфейс користувача): дозволяє студентам створювати власні схеми, перетягуючи компоненти (наприклад, резистори, конденсатори тощо) та налаштовувати прилади;

– сервер вимірювань: запрограмований на C++, отримує запити клієнтів та спирається на список дозволених комбінацій, що представляють схеми, які можуть бути побудовані, для їх перевірки або відхилення. Якщо результат перевірено, він визначає комбінацію реле для динамічної побудови схеми та пересилає її на сервер обладнання;

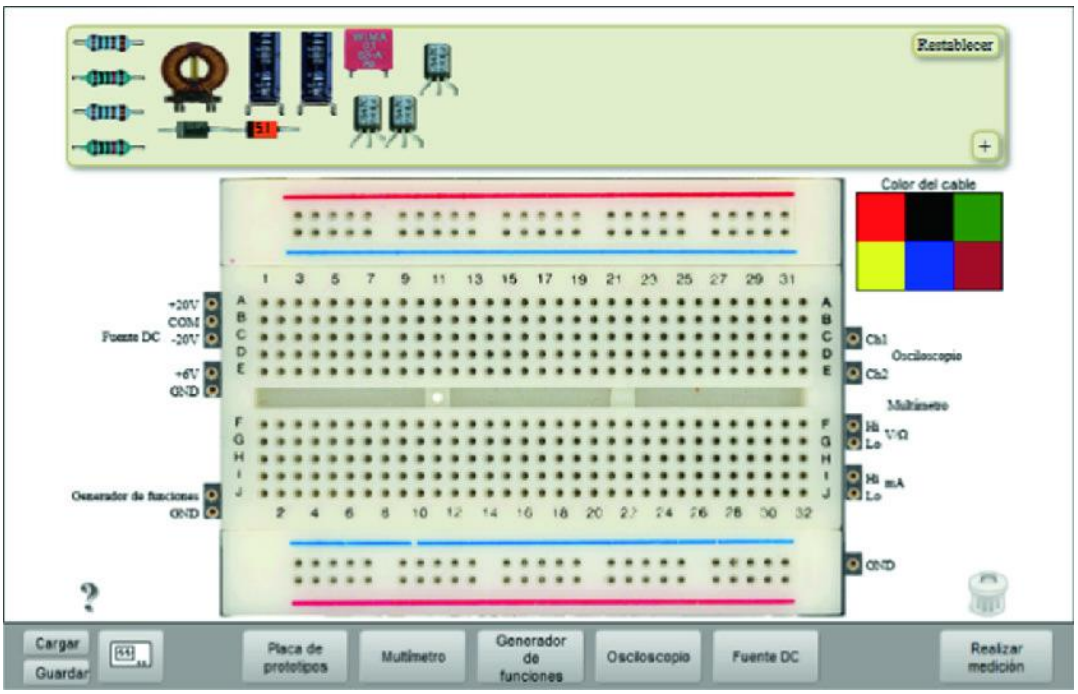


Рисунок 1.7 – Інтерфейс VISIR [21]

– сервер обладнання: на базі LabView, отримує перевірені запити від сервера вимірювань та фізично будує схеми, налаштовуючи обладнання National Instruments PXI та правильно встановлюючи реле в матриці комутації, керованій МК Microchip PIC.

Крім того, TUDO роками співпрацює з LabsLand, що надає віддалені експерименти та широкі можливості моделювання різноманітних процесів [22].

1.3 Висновки до розділу 1

Як показали проведені дослідження, кожна віддалена лабораторія має свої архітектурні та дидактичні особливості. Результати порівняльного аналізу розглянутих лабораторій наведено в табл. 1.1 [3], [18], [19], [21].

Зокрема, платформа Edrys зосереджена на децентралізованому доступі та створенні інтерактивних текстових курсів. VISIR надає спеціалізований інструментарій для вивчення електроніки, дозволяючи працювати з реальним обладнанням через зручну віртуальну макетну плату.

Таблиця 1.1 – Порівняльний аналіз віддалених лабораторій [3], [18], [19], [21]

Критерій порівняння	Edrys	GOLDi 2.0	VISIR
Концепція та тип лабораторії	Платформа дистанційного навчання з відкритим кодом для створення віддалених мікролабораторій	Гібридна лабораторія, що поєднує віртуальний, віддалений доступ та концепцію Hybrid Take-Home-Lab	Віддалена лабораторія для проведення фізичних експериментів з електроніки
Апаратне забезпечення експериментів	Симульовані та реальні плати Arduino, локальні сервери (наприклад, Raspberry Pi), вебкамери	Електромеханічні системи (ліфт, 3-осьова модель), мікроконтролери, ПЛІС (FPGA)	Обладнання National Instruments PXI, реле, матриці комутації, електронні компоненти (резистори, конденсатори)
Архітектура та організація доступу	Використання веббраузера як станції та ретранслятора, система комунікації на базі публікації/підписки	Гнучка грід-структура, що дозволяє підключати власні домашні блоки керування до складних університетських моделей	Трирівнева архітектура: вебінтерфейс (клієнт), сервер вимірювань (на C++) та сервер обладнання (на LabView)
Особливості навчального середовища	Використання модуля LiaScript для створення інтерактивних курсів із симуляціями безпосередньо в браузері	Можливість керування об'єктами з різних місць, паралелізація експериментів та інтеграція цифрових двійників	Графічний інтерфейс (GUI), який візуально відтворює макетну плату для інтуїтивної побудови схем

Натомість гібридна лабораторія GOLDi 2.0 пропонує найбільш комплексний підхід, поєднуючи віртуальні симуляції, віддалений доступ до складних електромеханічних моделей та концепцію Hybrid Take-Home-Lab.

Отже, проблему повної гнучкості та міжуніверситетської доступності цифрових лабораторій поки не вирішено остаточно. Тому актуальною задачею є подальший розвиток дидактичних, технічних та організаційних рішень (зокрема на базі гібридних грід-структур), що дозволять створити відкрите навчальне середовище, яке можна легко комбінувати та адаптувати до вимог навчання і потреб студентів.

2 ДОСЛІДЖЕННЯ МОДЕЛЕЙ І СПОСОБІВ ОПРАЦЮВАННЯ ДАНИХ ПРИ ВЗАЄМОДІЇ З ВІДДАЛЕНОЮ ЛАБОРАТОРІЄЮ

2.1 Моделі декомпозиції експерименту

Take-Home-Lab кейс – це новий лабораторний пристрій у розподіленій інфраструктурі гібридної лабораторії GOLDi 2.0, що забезпечує підключення до онлайн-лабораторії GOLDi через Інтернет (рис. 2.1) [19], [23].

Рисунок 2.1 – Інфраструктура GOLDi 2.0 [23]

Він надає інтерфейс з усіма входами та виходами фізичної системи у гібридній лабораторії GOLDi 2.0 (ліфт, 3-осьовий портал, склад з високими стелажами та інші). Кейс може використовуватися як інтерфейс ВЛ для її блоків керування, наприклад, платформ Arduino або Raspberry Pi, демонстраційних плат FPGA або наборів для цифрових логічних експериментів. Кейс також має макетну плату та з'єднувальні дроти, тому можна гнучко зібрати на ньому власну схему [23].

Наприклад, необхідно розробити алгоритм керування для переміщення каретки 3-осьового порталу вздовж траєкторії в площині (в межах прямокутної області) з двома ступенями свободи (ліворуч/праворуч та вгору/вниз). Позиціонування можливе за допомогою різних двигунів у кожному напрямку. Доступні чотири кінцеві вимикачі для виявлення прибуття каретки у відповідні кінцеві положення (ліворуч/праворуч/вгору/вниз) [18].

Метамодель декомпозиції експерименту наведено на рис. 2.2.

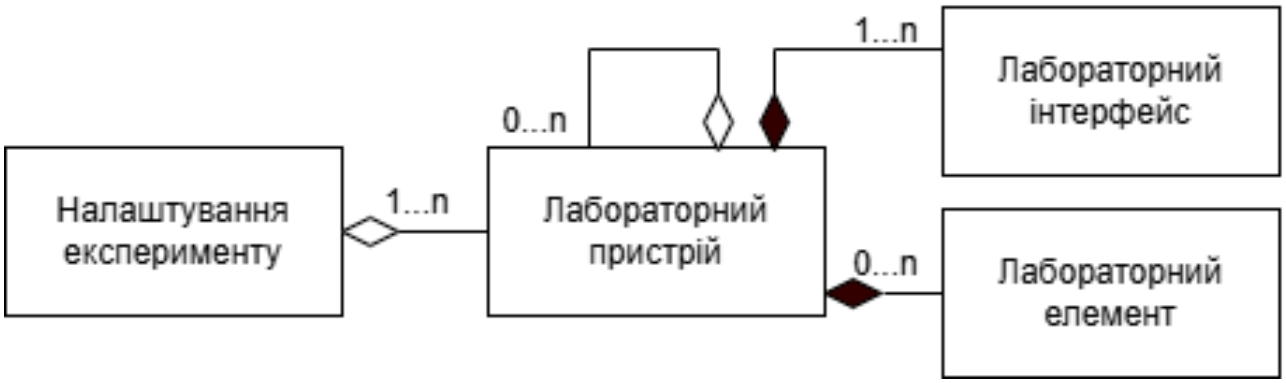


Рисунок 2.2 – Метамодель декомпозиції експерименту [18]

Модель декомпозиції експерименту показано на рис. 2.3. Лабораторними елементами є: двигуни, кінцеві вимикачі, інкрементальні енкодери, FPGA, МК, кнопки, перемикачі, логічні вентилі та дроти, а також сам користувач.

Різні лабораторні елементи можна об'єднати в базові модулі – лабораторні пристрої. У цьому лабораторному експерименті це плата підключення, модуль з інтерфейсом до портального крана, різні модулі керування (МК, PLD, PCB) (рис. 2.4), модуль перемикача та модуль користувача.

Кожен лабораторний пристрій містить один або декілька інтерфейсів. У цьому експерименті це тактильний/матеріальний інтерфейс між користувачем і кнопками/перемикачами, контакти як інтерфейс між макетною платою та різними модулями, а також з'єднувальні дроти як інтерфейс між модулями [18].

2.2 Сервісно-орієнтована архітектура віддаленої лабораторії

Як показали проведені дослідження, IEEE 1876 – це стандарт для мережевих інтелектуальних навчальних об'єктів для онлайн-лабораторій [24], що спеціально розроблений для забезпечення сумісності між кількома ВЛ.

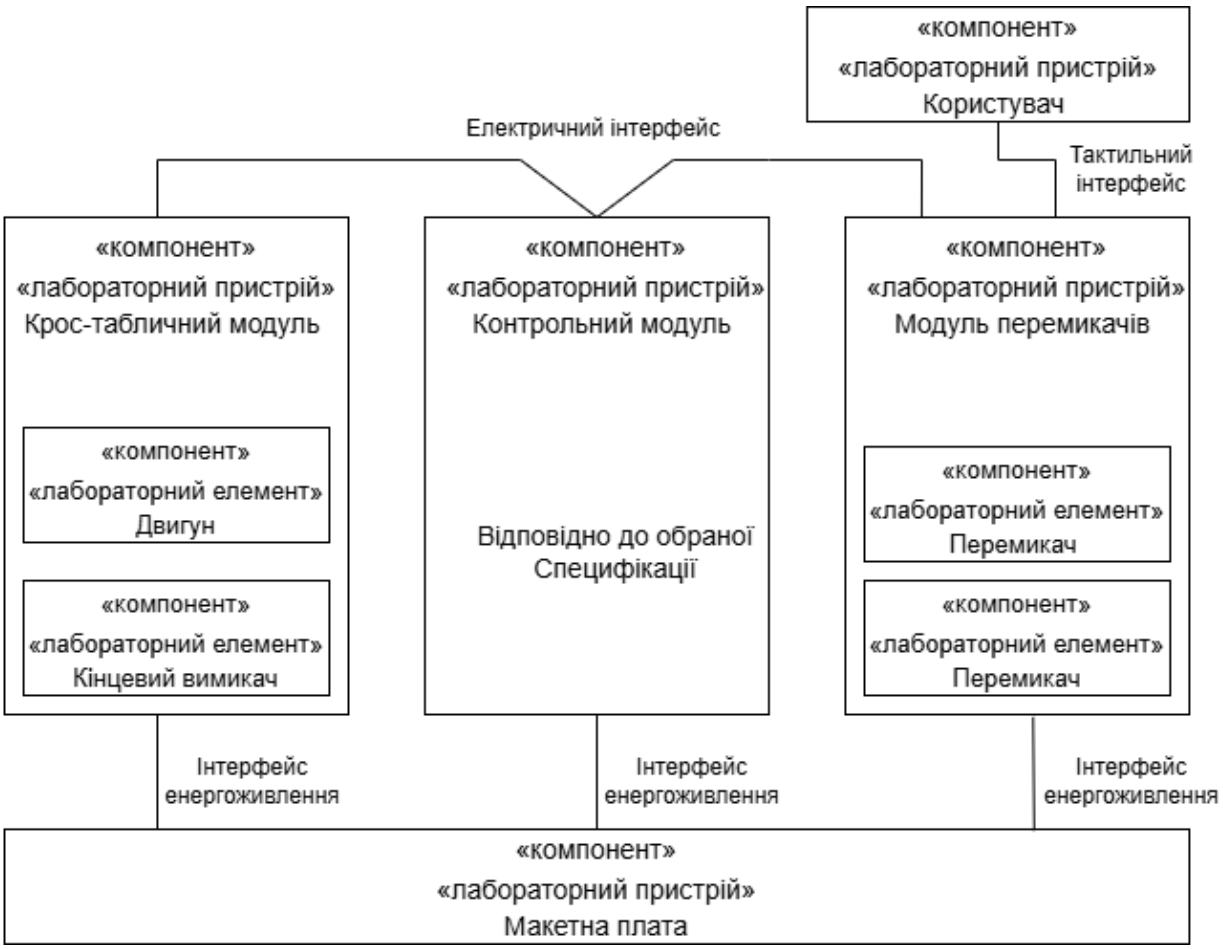


Рисунок 2.3 – Модель декомпозиції лабораторного експерименту [18]

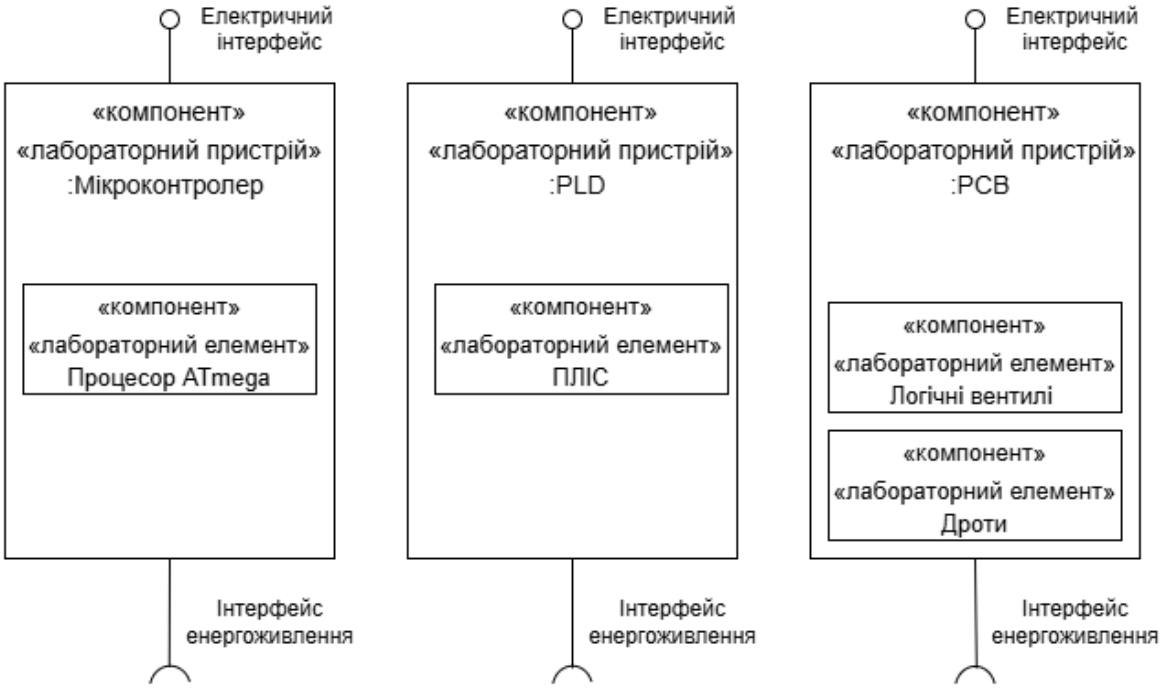
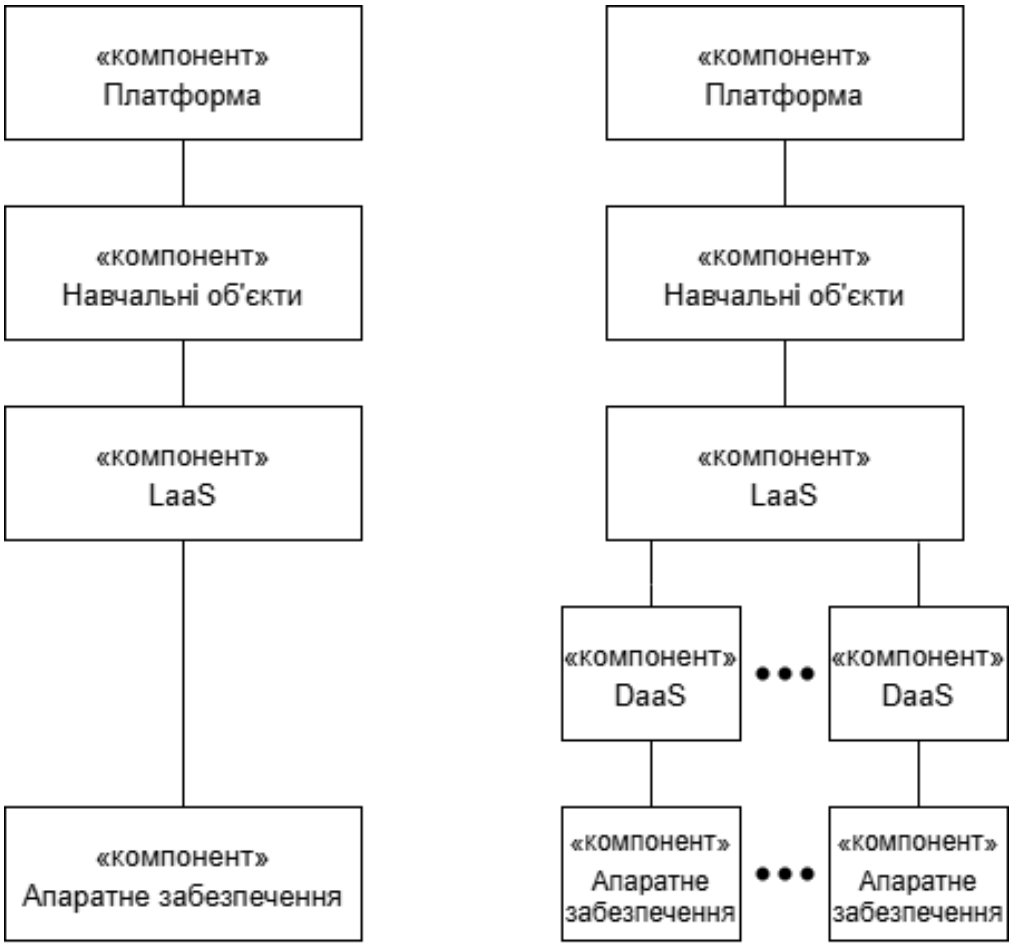


Рисунок 2.4 – Варіанти модулів керування експериментом [18]

Він базується на багаторівневій архітектурі ВЛ, що складається з апаратного забезпечення, Лабораторії як Послуги (Lab as a Service, LaaS), навчальних об'єктів та платформного рівня (рис. 2.5, а). Стандарт визначає інтерфейс LaaS та профіль програми, адаптований для ВЛ згідно з стандартом IEEE щодо метаданих навчальних об'єктів [25].



а) б)

а – архітектура, запропонована в стандарті IEEE 1876; б – архітектура адаптована для ВЛ

Рисунок 2.5 – Варіанти багаторівневої архітектури [18]

Практична реалізація на основі цього стандарту є складною, оскільки він не розрізняє апаратне забезпечення та об'єкти взаємодії з користувачем, які обидва розглядаються як лабораторні пристрої. Однак можливо

інкапсулювати всю цю складність на рівні LaaS та запропонувати попередньо налаштовані експерименти, хоча гнучкість побудови експериментів з компонентів не буде забезпечено. Цей підхід повністю сумісний зі стандартом, як показано на рис. 2.5, б, де лабораторні пристрої пропонуються як послуга та відображаються як Пристрій як Послуга (Device as a Service, DaaS) [18].

Сервісно-орієнтована архітектура (Service-Oriented Architecture, SOA) розглядається як життєздатне рішення для декомпозиції експерименту. Інші рішення, зокрема клієнт-серверна архітектура, погано відповідають сервісам, що може пропонувати лабораторний об'єкт, які у випадку електричного з'єднання не є спрямованими та дуже чутливі до затримки. При цьому доцільно розглядати дві частини архітектури: одна дозволяє налаштовувати та ініціалізувати експеримент, а інша працює під час виконання експерименту [18].

Усі лабораторні пристрої розроблені відповідно до принципів сервісно-орієнтованої архітектури. Це означає, що кожен лабораторний пристрій пропонує та споживає різні сервіси. Прикладами є такі.

Сервіс вебкамери. Відеопотоки передаються між окремими лабораторними пристроями через цей сервіс. Наприклад, 3-осьовий портал моделі електромеханічного обладнання передає відеопотік, а інтерфейс користувача (Experiment Control Panel, ECP) сприймає його для відображення в браузері.

Сервіс передачі файлів. МК або FPGA вибрано як лабораторний пристрій для керування моделями обладнання, програмний файл розробленого алгоритму керування (*.hex, *.pof) може бути переданий на нього через сервіс передачі файлів для подальшого програмування.

Сервіс електричного з'єднання. Цей сервіс визначає, які сигнали датчиків або виконавчих механізмів лабораторного пристрою (наприклад, 3-осьового порталу) підключені до яких електричних сигналів іншого лабораторного пристрою для заданого експерименту. Тут надаються різні

підсервіси. У найпростішому випадку цифрові з'єднання встановлюються через підсервіс GPIO. Однак, також можливо реалізувати з'єднання на основі протоколів через підсервіси, як I2C, SPI або PWM [26].

На рис. 2.6 представлено мікросервісну архітектуру гібридної лабораторії GOLDi 2.0, інтегрованої з LMS Moodle [18].

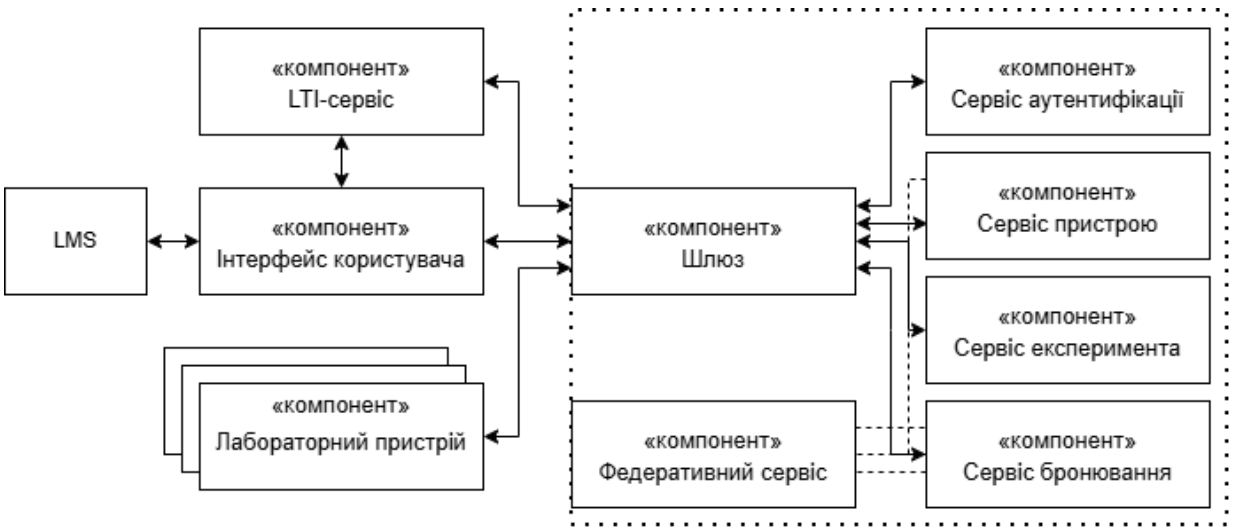


Рисунок 2.6 – Мікросервісна архітектура гібридної лабораторії [18]

Компонент LTI (Learning Tools Interoperability) – це технічний стандарт, який дозволяє Moodle безпечно підключатися до зовнішніх програм та постачальників контенту [27].

LTI 1.3 та LTI Advantage, розроблені 1EdTech, представляють найновіші, найбезпечніші та найефективніші стандарти для підключення зовнішніх навчальних інструментів до систем управління навчанням (Learning Management System, LMS). Вони використовують вебтокени OAuth2 та JSON для підвищення безпеки, а Advantage надає три ключові послуги: глибоке посилання, послуги призначення та оцінювання, а також надання імен та ролей [27].

Ключові функції LTI у Moodle є такими:

– безшовна інтеграція: LTI дозволяє додавати сторонні інструменти безпосередньо до курсу Moodle та робити їх відображуваними як нативні завдання;

– єдиний вхід (SSO): користувачам не потрібно окремо входити до зовнішнього інструменту; посилання LTI автоматично автентифікує їх, заощаджуючи час;

– повернення оцінок: LTI дозволяє зовнішнім інструментам надсилати оцінки назад до журналу оцінок Moodle, автоматизуючи звітність;

– передача даних: LTI безпечно передає необхідну інформацію (наприклад, ідентифікатор користувача, роль та контекст курсу) між Moodle та зовнішнім застосунком [27].

Moodle може працювати у двох ролях LTI:

– споживач інструменту LTI (клієнт): Moodle посилається на зовнішній інструмент. Викладач додає в Moodle дію «Зовнішній інструмент», яка посилається на сторонній застосунок;

– постачальник інструменту LTI (сервер): Moodle надає власні курси або дії для використання на іншій платформі (наприклад, на іншому сайті Moodle або іншій LMS) [27].

2.3 Аналіз процесу опрацювання даних при взаємодії з віддаленою лабораторією

Процес опрацювання даних при взаємодії з ВЛ складається з концептуального формулювання та реалізації алгоритму керування за допомогою різних систем розробки для досягнення остаточного валідованого результату. Його можна завантажити через панель керування експериментом (ECP), що працює у веббраузері, на вибраний блок керування у ВЛ, який буде автоматично запрограмований у фоновому режимі, після чого експеримент виконується дистанційно. Для цього користувачу GOLDi доводиться використовувати додаткові сторонні інструменти на своєму персональному комп'ютері (ПК) (рис. 2.7) [28].

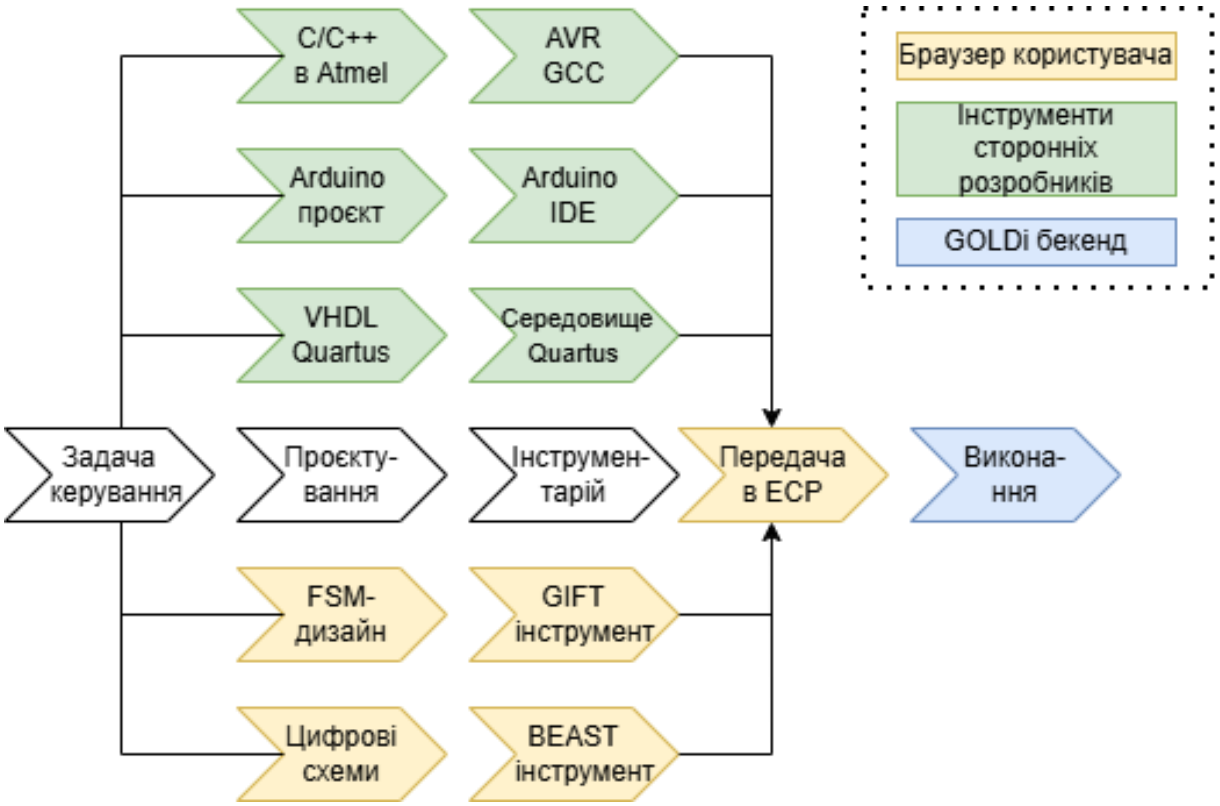


Рисунок 2.7 – Схема традиційного процесу опрацювання даних [28]

Для програмно-орієнтованої реалізації користувачі можуть реалізувати свій алгоритм керування безпосередньо для МК. У GOLDi підтримується створення ПЗ для МК з використанням зовнішніх безкоштовних засобів розробки (наприклад, Atmel Studio [29], Arduino IDE [30]) для введення коду C/C++ або Arduino та для компіляції проєкту. Згенерований файл прошивки необхідно завантажити через ECP до системи GOLDi, яка автоматично запрограмує МК у фоновому режимі (рис. 2.8) [28].

Основним недоліком традиційного процесу опрацювання даних є використання зовнішніх інструментів розробки від сторонніх виробників. Встановлення та налаштування цих інструментів на ПК або ноутбуці студента може бути досить складним для початківців.

Також можуть виникнути проблеми з обмеженим дисковим простором, несумісними операційними системами або непродуктивними ПК.

Оскільки ці інструменти орієнтовані на інженерів та розробників ПЗ, які вже знайомі з процесом розробки, вони зазвичай мають багато

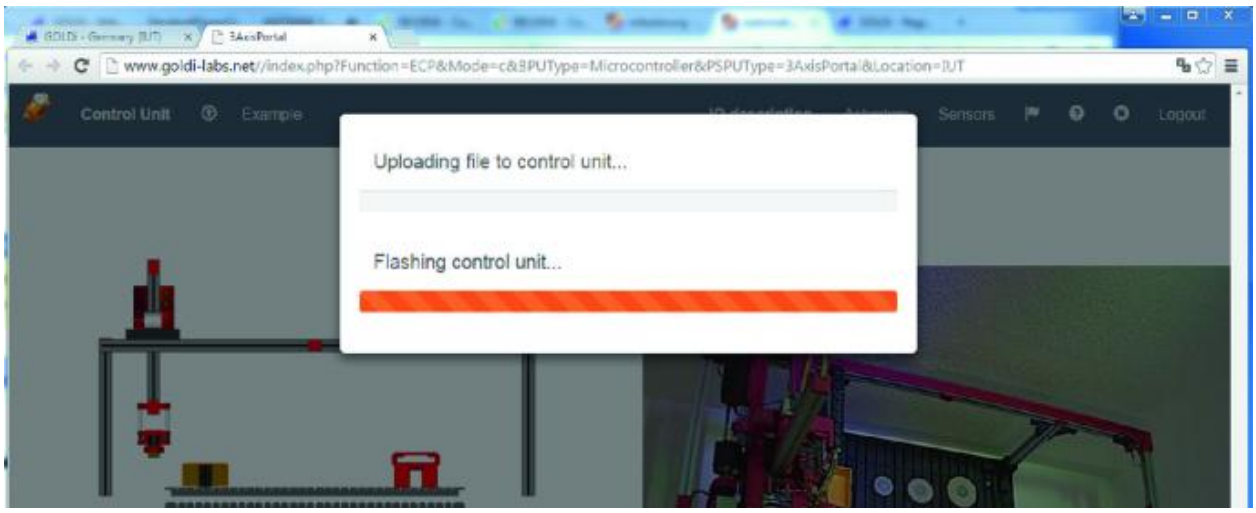


Рисунок 2.8 – Програмування МК через ECP [28]

додаткових функцій та налаштувань, які новачок не може інтуїтивно зрозуміти. Загалом це призводить до дуже високого вхідного бар'єру, особливо для студентів молодших курсів [28].

Тому оновлений процес опрацювання даних спрямований на два аспекти: простий та інтуїтивно зрозумілий інтерфейс користувача та скорочення кількості ПЗ, яке користувач має встановлювати на своєму ПК [28].

Для цього розроблено вебсистему, що дозволяє зменшити зусилля студента, починаючи від пошуку потрібного ПЗ, завантаження та встановлення програмного пакета та закінчуючи відвідуванням вебсайту лабораторії. Цей підхід також реалізовано, наприклад, у [31], але він доступний лише для апаратно-орієнтованого підходу в VHDL для FPGA.

На рис. 2.9 наведено удосконалену схему процесу опрацювання даних, особливістю якої є відсутність сторонніх інструментів, оскільки їх перенесли з ПК користувачів на хмарний сервер. Тепер користувач має один уніфікований інструмент WIDE (Web IDE).

Це дозволяє знизити бар'єр від зміни мов та середовищ програмування, оскільки середовище користувача залишиться тим самим,

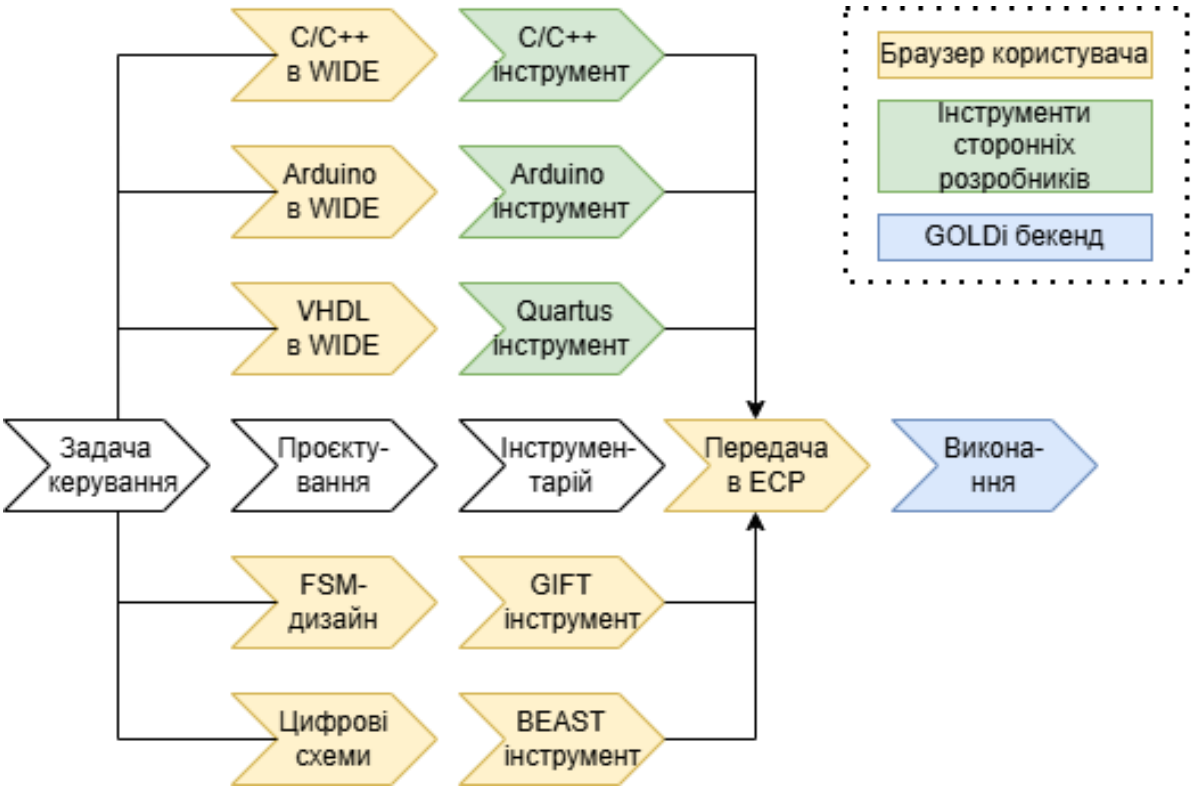


Рисунок 2.9 – Удосконалена схема процесу опрацювання даних [28]

за винятком мови, яка використовуватиметься під час процесу проєктування [28].

Важливим аспектом використання системи WIDE в інфраструктурі GOLDi є те, що WIDE має бути призначений певному експерименту, тобто комбінації блоку керування та фізичної системи. Для цього реалізовано два режими роботи для системи WIDE:

– автономний режим, де користувач може працювати над своїм проєктом незалежно від будь-якого резервування експерименту. При цьому користувачеві необхідно вибрати правильну мову програмування та налаштування експерименту перед розробкою в цьому режимі;

– режим роботи, що пов’язаний та інтегрований з ECP. У цьому режимі користувач може завантажувати свій проєкт безпосередньо до вибраного блоку керування та тестувати його з фізичною системою. На відміну від автономного режиму, цей режим обмежений часом роботи в декілька хвилин, щоб заохотити користувача розробляти своє рішення, використовуючи автономну версію [28].

2.4 Моделювання апаратного забезпечення експериментів

У віддаленому лабораторному середовищі GOLDi [32] 3-осьовий портал та портальний кран можуть розглядатися як різні типи автоматизованих промислових систем обробки, що використовуються для навчання програмуванню програмованих логічних контролерів та мехатроніки, які відрізняються переважно своєю кінематичною структурою та типовими сценаріями застосування. Результати порівняння цих експериментів наведено в табл. 2.1.

2.4.1 Модель портального крана

Проєктне завдання для подання даних моделі портального крана формулюється так [32].

На шпинделі портального крана інструментальна каретка може переміщуватися праворуч і ліворуч. Кінцеві вимикачі надають вхідну інформацію про ліве кінцеве положення (*xl*), а також праве кінцеве положення (*xr*) інструментальної каретки (*xr*, *xl*).

Рухом можна керувати за допомогою бінарних вихідних змінних (*yl*, *yr*):

– рух ліворуч (*yl* = 1, *yr* = 0);
– рух праворуч (*yl* = 0, *yr* = 1);
– зупинка (*yl* = *yr* = 0).

Додаткова вхідна змінна *xs* сигналізує про:

– зупинку руху (*xs* = 0);
– рух (*xs* = 1) праворуч або ліворуч.

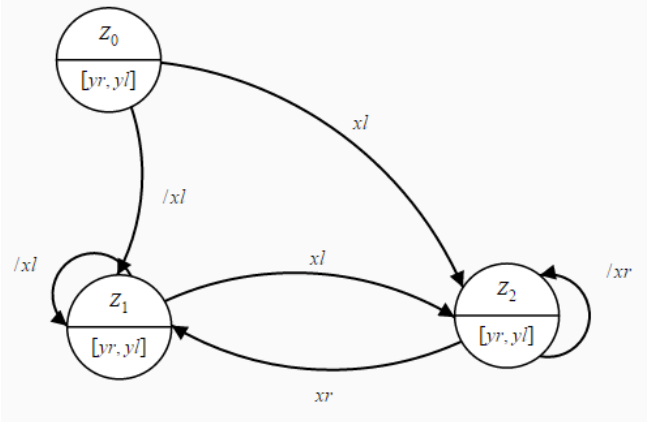
Після можливої паузи потрібно продовжити рух в початковому напрямку. Відповідно, ініціалізація (*yl* = 0, *yr* = 0), рух ліворуч (*yl* = *xs*, *yr* = 0), рух праворуч (*yl* = 0, *yr* = *xs*).

Таблиця 2.1 – Порівняльна характеристика експериментів

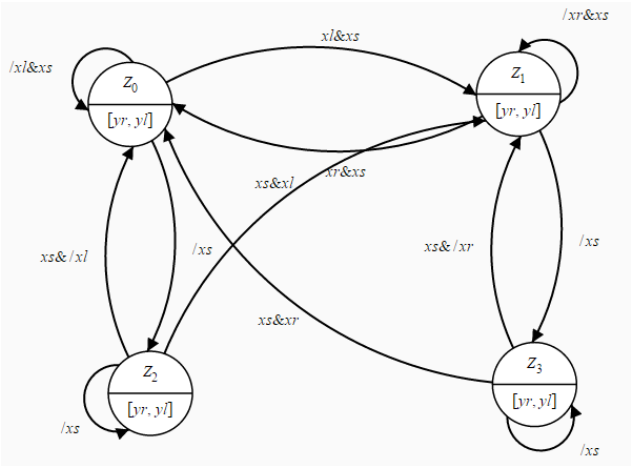
Експе-римент	Опис	Конфігурація та рух осей	Тип та застосування захоплення	Точність та керування
3-х осьовий портал	Високоточний декартов робот з трьома лінійними осями	Складається з трьох лінійних осей (X, Y, Z), які забезпечують рух інструментальної каретки у трьох напрямках. Оснащений електромагнітним захоплювачем, який рухається вздовж мосту крана (Y), причому сам міст рухається вздовж рейок (X), а захоплювач рухається вертикально (Z)	Використовує електромагнітне захоплення та часто використовується для моделювання стаціонарного робота для переміщення заготовок до обробного блоку в рамках моделювання автоматизованого виробництва	Зосереджений на точному позиціонуванні X-Y-Z для переміщення деталей між «складом деталей» та «станцією розвантаження». Він використовує програмні та апаратні кінцеві вимикачі для визначення свого робочого простору
Портальний кран	Конструкція портального типу, призначена для обробки/ підйому	Проєктується як 2- або 3-осьова система, що фокусується на горизонтальному транспортуванні (портальний рух) та вертикальному підйомі (підйомник), призначена для переміщення по великих відкритих площах або транспортування сировини	Оснащений захоплювачем для гранул або механічним підйомником, оптимізований для забирання матеріалів, їх транспортування до проміжного сховища та скидання їх у контейнери, що часто зустрічається в промислових сценаріях обробки	Робить акцент на високій вантажопідйомності та охопленні великої площі, іноді з меншою точністю позиціонування порівняно з 3-осьовим роботом, і використовується для завантаження, розвантаження та штабелювання

Модель у вигляді графа скінченного автомата Мілі наведено на рис. 2.10, а. Це орієнтований граф, що візуалізує скінченний автомат (Finite State Machine, FSM), у якому вихідні сигнали залежать як від поточного стану автомату, так і від вхідних значень [32].

Модель у вигляді графа скінченного автомата Мура наведено на рис. 2.10, б. Це орієнтований граф, що візуалізує скінченний автомат, у якому вихідні сигнали залежать лише від поточного стану автомату та не залежать від вхідних значень [32].



а)



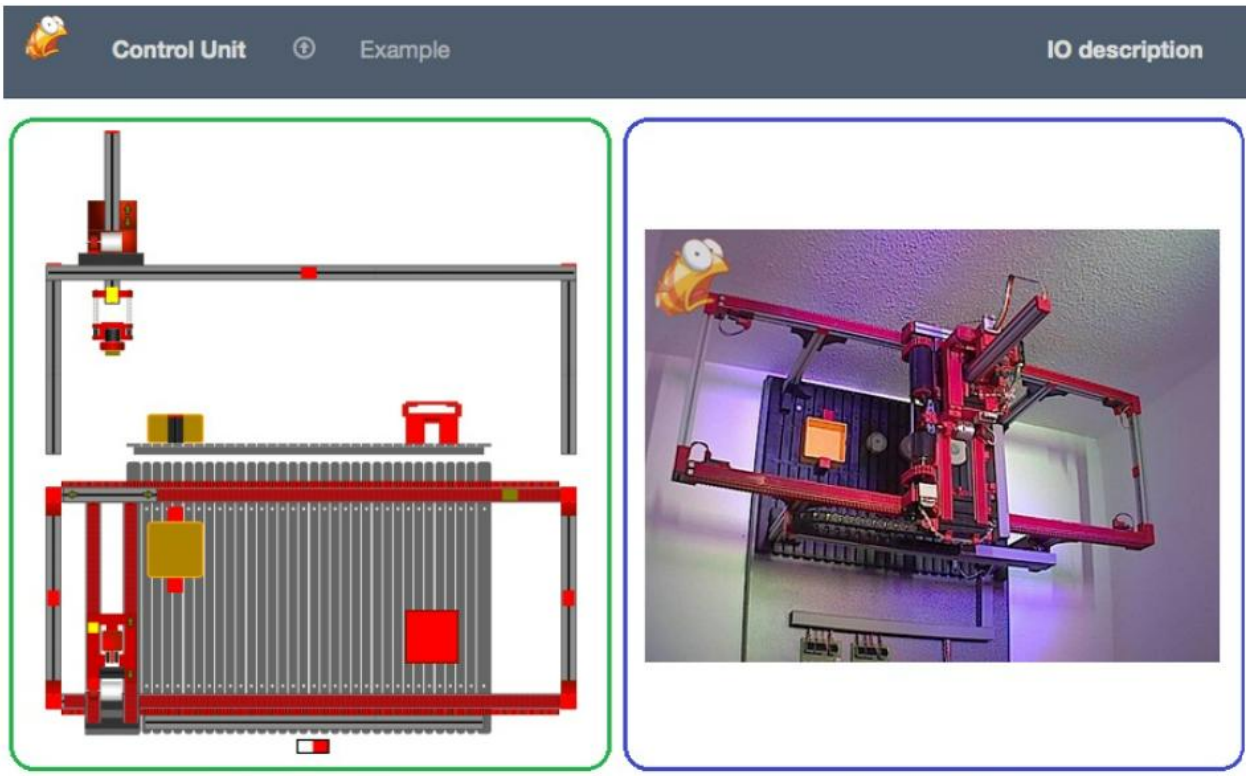
б)

а – модель у вигляді графа автомата Мілі; б – модель у вигляді графа автомата Мура

Рисунок 2.10 – Моделі портального крана [32]

2.4.2 Модель 3-х осьового порталу

На рис. 2.11 показано експеримент 3-осьовий портал як віртуальну модель (а) та фізичну систему (б). Каретку можна рухати по осі X та по осі Y, його електромагнітний захоплювач можна рухати в напрямку Z. Крім того, магніт захоплювача можна вмикати та вимикати [32].



а) б)

а - віртуальна модель; б - фізична система

Рисунок 2.11 – Експеримент 3-осьовий портал [32]

Проєктне завдання для подання даних моделі 3-х осьового порталу формулюється так [32].

Розробити алгоритм керування інструментальною кареткою 3-х осьового порталу з двома ступенями свободи, який:

– створює чотири сигнали керування двигуном для бажаних напрямків вліво/вправо/вгору/вниз (*yl*, *yr*, *yu*, *yo*);

– аналізує чотири кінцеві вимикачі вліво/вправо/вгору/вниз (*xl*, *xr*, *xu*, *xo*).

Це відбувається відповідно до такої послідовності:

– інструментальна каретка повинна рухатися якомога швидше (незалежно від її початкового положення) після 1/0-фронту стартового сигналу (*xs*) до лівого/нижнього кута;

– після цього вона повинна рухатися вгору до лівого/верхнього кута;

– та вздовж верхньої межі вправо, доки не досягне правого/верхнього кута, де рух має бути зупинений.

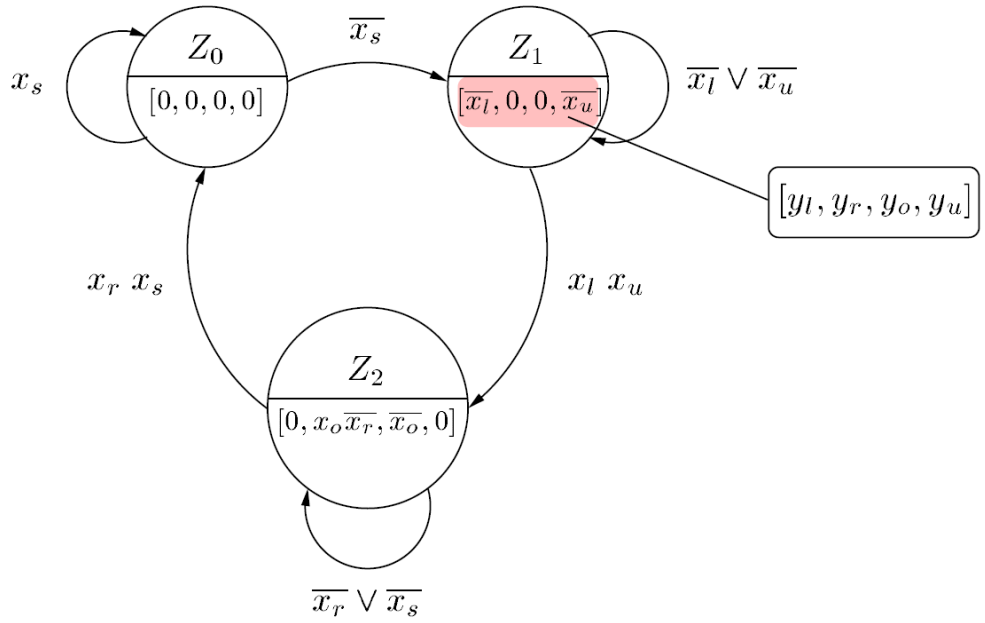
Перезапуск руху можливий лише за допомогою ще одного 1/0-фронту для *xs*.

Модель у вигляді графа скінченного автомата Мілі наведено на рис. 2.12, а, у вигляді графа скінченного автомата Мура – на рис.2.12, б [32].

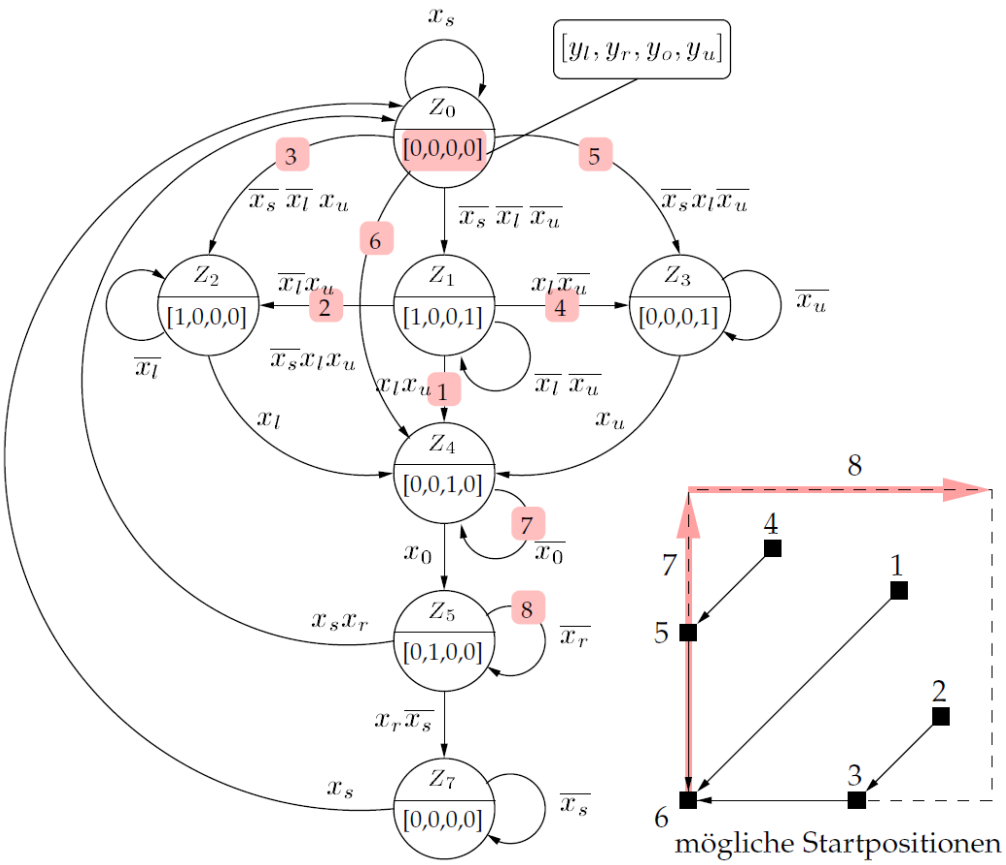
2.5 Постановка завдань дипломної роботи

Як показали проведені дослідження, програмна реалізація алгоритмів керування для віддалених експериментів лабораторії GOLDi 2.0 може бути реалізована з використанням кейсу Take-Home-Lab на основі плати Arduino або інших плат, МК та додаткових електронних компонентів. При цьому можливо використовувати як безкоштовне середовище Arduino IDE, так і веборієнтоване середовище WIDE. Але питання підвищення залученості користувача для взаємодії з інфраструктурою гібридної лабораторії GOLDi 2.0 залишається актуальним.

Мета роботи – покращення користувацького досвіду при взаємодії з віддаленою лабораторією за рахунок моделей та способів інтегрованого опрацювання даних.



а)



б)

а – модель у вигляді графа автомата Мілі; б – модель у вигляді графа автомата Мура

Рисунок 2.12 – Моделі 3-х осьового порталу [32]

Для досягнення мети роботи необхідно виконати такі завдання:

– виконати аналіз вимог до розробки, розробити архітектуру системи та моделі взаємодії з віддаленою лабораторією;

– обрати інструментарій та технології реалізації програмно-апаратного забезпечення для взаємодії з віддаленою лабораторією;

– розробити узагальнений алгоритм програмно-апаратної взаємодії з віддаленим експериментом;

– розробити програмне забезпечення для інтегрованого опрацювання даних віддаленого експерименту в інфраструктурі гібридної лабораторії GOLDi 2.0;

– виконати порівняльний аналіз реалізованих способів опрацювання даних при взаємодії з віддаленою лабораторією.

3 ПРОЄКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Аналіз вимог

Вимоги до розробки:

– створити програмно-апаратну систему для керування 3-осьовим порталом, підключеним через GPIO;

– використовувати концепції та конструкції мови програмування C++ для реалізації ПЗ;

– структурувати код C++ так, щоб його було легко тестувати та підтримувати в довгостроковій перспективі;

– визначати відповідні структури даних для керування портальним краном та правильно їх використовувати;

– використовувати інтерфейси низькорівневої розробки ПЗ, такі як інтерфейси драйверів операційної системи;

– використовувати програмний інтерфейс GPIO.

Програмний інтерфейс GPIO (General Purpose Input/Output) – це цифровий API, який дозволяє ПЗ налаштовувати, зчитувати та записувати дані на фізичні контакти МК або процесора. Він діє як посередник між кодом програми та апаратним забезпеченням, що дозволяє керувати периферійними пристроями, такими як світлодіоди, кнопки та датчики. Звичайні операції включають встановлення напрямку контактів (введення/виведення), зчитування рівнів входу та керування тригерами переривань [33]-[35].

Ключові аспекти програмних інтерфейсів GPIO такі.

Конфігурація: контакти можуть бути налаштовані як входи (наприклад, для зчитування перемикачів) або виходи (наприклад, для керування світлодіодами) з внутрішніми підтягувальними/знижувальними резисторами.

Керування/доступ: функції API дозволяють встановлювати цифровий стан (високий/низький) або зчитувати поточний стан напруги.

Переривання: ПЗ може реєструвати зворотні виклики для виявлення фронтів (зростаючих або спадаючих фронтів).

Абстракція: API, такі як у драйверах ядра Linux, дереві пристроїв Zephyr RTOS або Python на Raspberry Pi, приховують складність програмування на рівні регістрів.

Модель драйвера: сучасні системи часто використовують дерева пристроїв для визначення ролей GPIO, забезпечуючи апаратно-незалежний підхід [33]-[35].

3.2 Архітектура системи

На рис. 3.1 представлено архітектуру ВЛ, яка ілюструє два альтернативні підходи до розгортання інфраструктури для виконання віддалених експериментів. Система об’єднує три основні інформаційні домени: локальне робоче місце користувача, комунікаційне середовище передачі даних та програмно-апаратне забезпечення ВЛ. Початковою точкою взаємодії виступає робочий ПК користувача, з якого ініціюється робота системи. Центральною ланкою маршрутизації на стороні апаратного комплексу є серверний вузол GOLDi Gateway, який виконує функцію шлюзу між глобальною мережею та фізичним обладнанням лабораторії.

Перший підхід (апаратно-орієнтований) реалізується через використання фізичного обладнання (кейсу Hybrid Take-Home-Lab) на стороні користувача. Програмний код завантажується через USB кабель на плату Arduino UNO, що є базовим обчислювальним ядром кейсу. Цей фізичний блок містить мережевий інтерфейс для зв'язку з GOLDi, а також додаткові компоненти (наприклад, кнопку для генерації інтерактивних переривань).

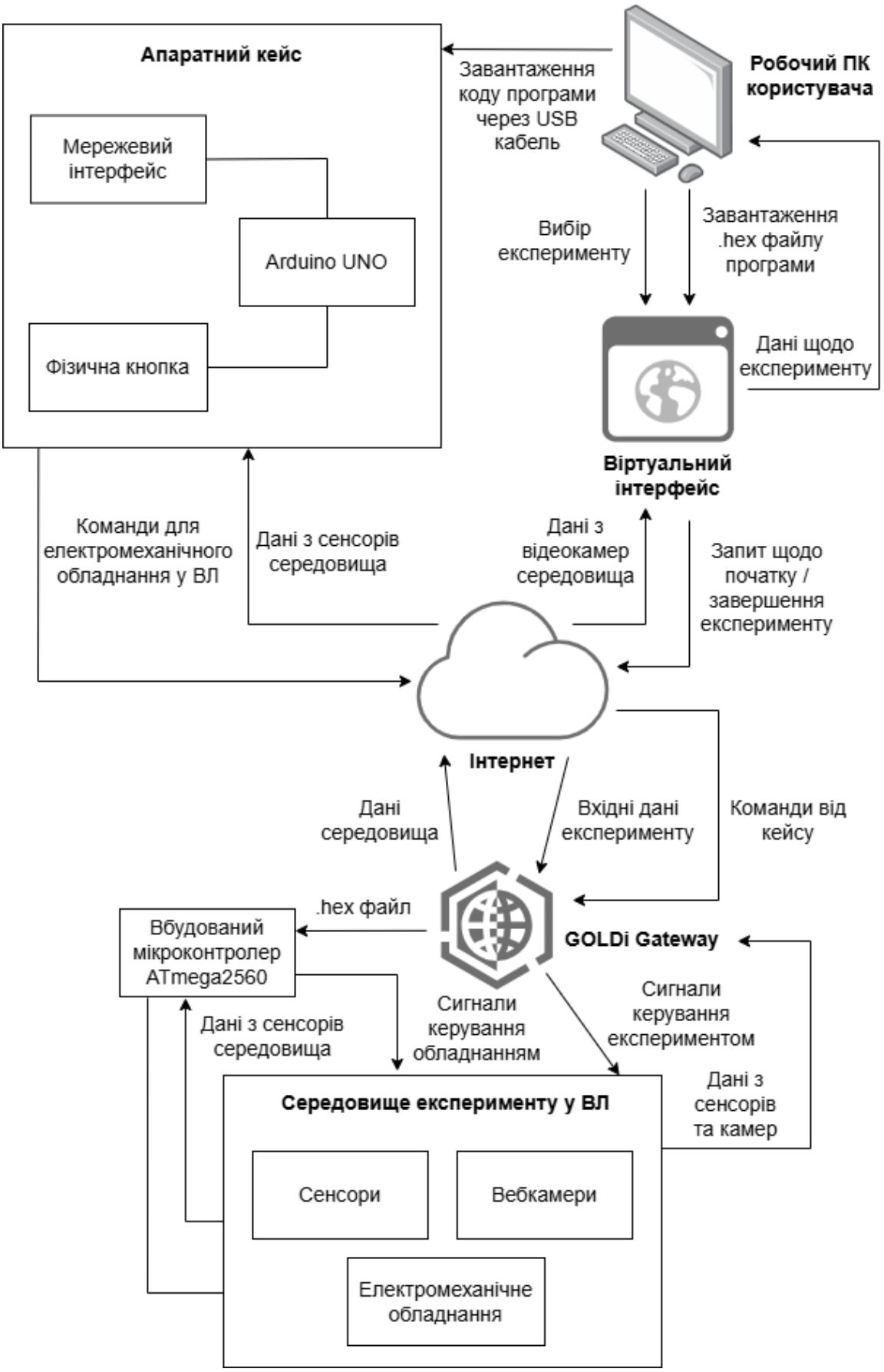


Рисунок 3.1 – Архітектура системи

Використання кейсу Hybrid Take-Home-Lab забезпечує гнучкість, оскільки користувач може додавати різні електронні компоненти, а також змінювати логіку та порядок підключення. Зокрема, взаємодія з фізичною кнопкою дозволяє запускати програму на виконання багаторазово без потреби перезавантаження експерименту. Команди для керування електромеханічним обладнанням у ВЛ генеруються локально МК на платі Arduino UNO та безперервно транслюються через Інтернет до GOLDi Gateway. Шлюз, у свою чергу, перетворює ці мережеві пакети на сигнали керування фізичним обладнанням експерименту. Одночасно з цим, дані з сенсорів середовища експерименту повертаються зворотним маршрутом до апаратного кейса, формуючи замкнений контур керування в режимі реального часу, який суттєво залежить від стабільності та пропускної здатності інтернет-з'єднання.

Другий підхід (віртуалізований) усуває потребу в локальному апаратному забезпеченні на стороні користувача. Взаємодія відбувається через програмний вебзастосунок – віртуальний інтерфейс. Користувач ініціює віддалене завантаження *.hex файлу програми для модуля керування на основі вбудованого МК ATmega2560. Цей скомпільований бінарний файл разом із запитом щодо початку або завершення експерименту передається для завантаження в МК модуля керування. Після успішної ініціалізації МК функціонує повністю автономно, генеруючи локальні сигнали керування обладнанням та зчитуючи дані з сенсорів середовища. Оскільки весь цикл опитування датчиків та видачі керуючих впливів відбувається безпосередньо на стороні ВЛ, це мінімізує мережеві затримки та суттєво підвищує загальну ефективність виконання експерименту, проте обмежує можливості користувача щодо конфігурування експерименту. Цей підхід також виключає будь-яку взаємодію користувача з обладнанням під час виконання експерименту, тож для повторного виконання експеримент необхідно перезавантажувати.

Незважаючи на відмінності у способах розгортання та логіки керування, цільовим об'єктом в обох сценаріях є середовище експерименту у ВЛ. Воно інтегрує в собі фізичне електромеханічне обладнання, набір позиційних сенсорів та вебкамери. Для забезпечення повноцінного зворотнього зв'язку та ефекту присутності, GOLDi Gateway безперервно транслює дані з відеокамер та сенсорів середовища назад до користувача. Це дозволяє користувачу отримувати об'єктивну інформацію щодо експерименту на свій робочий ПК для подальшого аналізу та верифікації розроблених алгоритмів.

3.3 Алгоритм програмно-апаратної взаємодії з експериментом

Схему алгоритму, що показує кроки необхідні для реалізації експерименту керування 3-осьовим порталом за допомогою програмно-апаратного забезпечення кейсу Take-Home-Lab, наведено на рис. 3.2. Цей алгоритм складається з таких етапів.

Блок 1. Початок.

Блок 2. Налаштування експерименту. Після входу на вебсайт GOLDi [35] користувач спочатку налаштовує експеримент. Для цього він вибирає лабораторні пристрої для експерименту зі списку доступних пристроїв.

Різні університети, що беруть участь у хмарі GOLDi, можуть надавати ці лабораторні пристрої. Кожен пристрій має унікальний ідентифікатор пристрою та токен входу й керується центральною службою пристроїв GOLDi. У цьому прикладі лабораторними пристроями, що використовуються, є модель електромеханічного обладнання 3-осьовий портал, лабораторний пристрій ECP як інтерфейс користувача та кейс Take-Home-Lab як пристрій керування. Однак, для експерименту можна використовувати більше лабораторних пристроїв.

Рисунок 3.2 – Схема алгоритму взаємодії з експериментом

Блок 3. З’єднання обраних сервісів (рис. 3.3) [23].

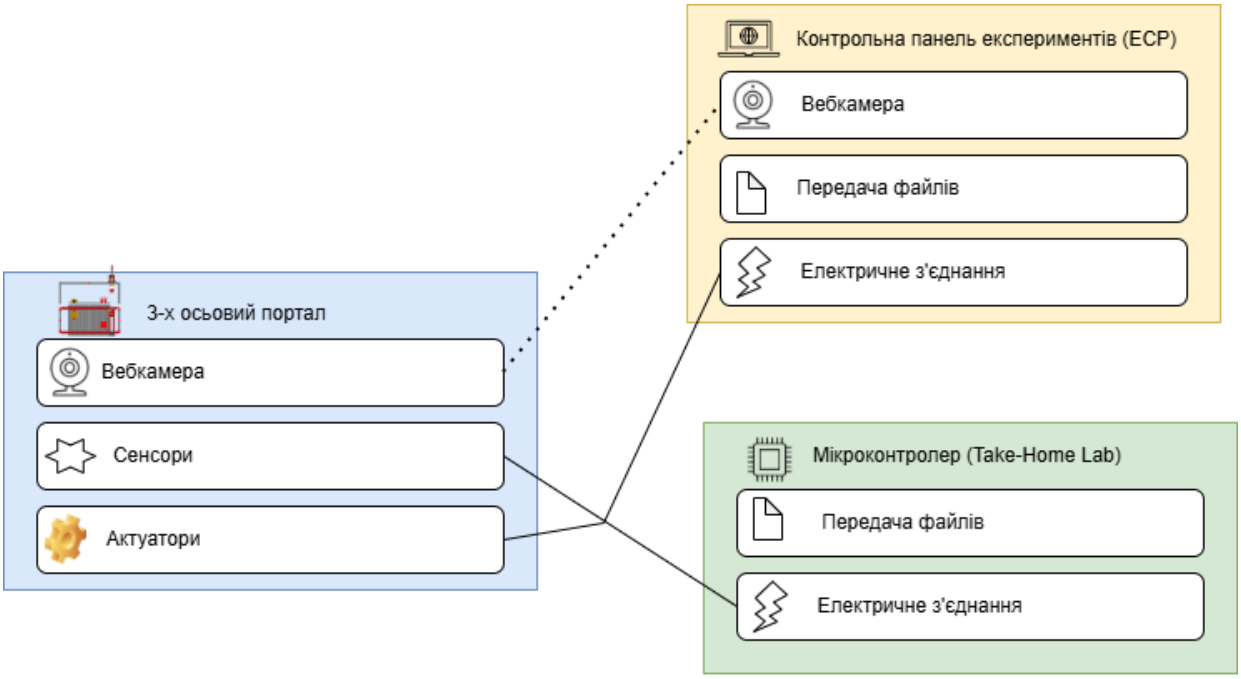


Рисунок 3.3 – Взаємозв’язок сервісів лабораторних пристроїв [23]

Лабораторний пристрій 3-осьовий портал пропонує сервіси вебкамери та електричного з'єднання (датчики та виконавчі механізми).

Лабораторний пристрій ECP пропонує вебкамеру, передачу файлів (не використовується для цього експерименту) та сервіс електричного з'єднання.

Кейс Take-Home-Lab пропонує послугу електричного з'єднання (рис. 3.4). Підключення двох служб вебкамери 3-осьового порталу та ECP забезпечує подальше спостереження за експериментом через вебкамеру. Сервіси електричного з'єднання повинні бути підключені до всіх трьох пристроїв (див. рис. 3.3).

Блок 4. Налаштування різних сервісів. Тут як приклад буде розглянуто лише службу електричного з'єднання. Планувальник сигналів/виводів відображає всі доступні сигнали для кожного пристрою (рис.3.5) [23].

Для 3-осьового порталу перелічено всі сенсори (*x_right*, *x_left*, ...) та актуатори (*engine_right*, *engine_left*, ...).

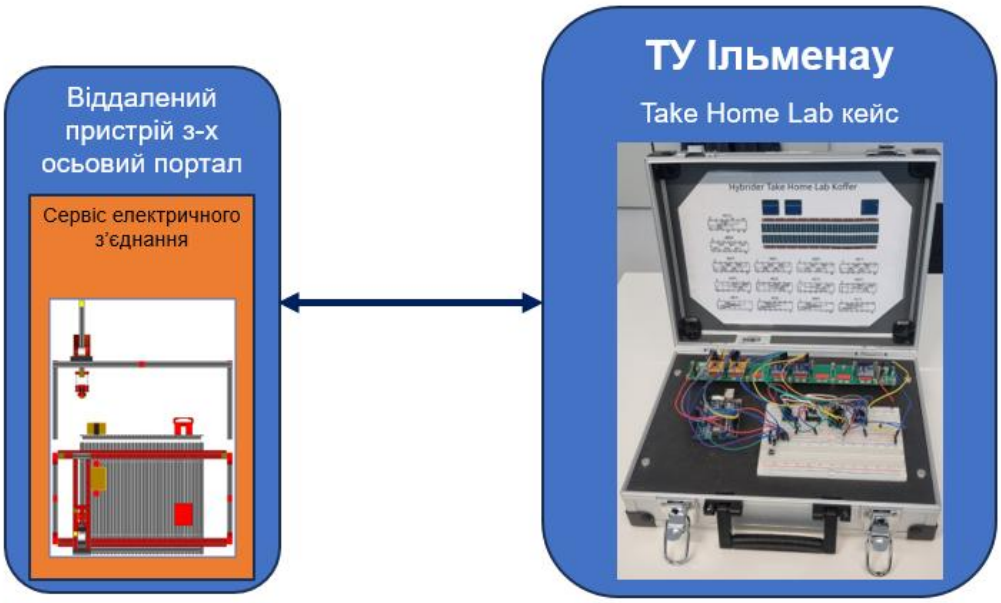


Рисунок 3.4 – Взаємодія Take-Home-Lab та експерименту

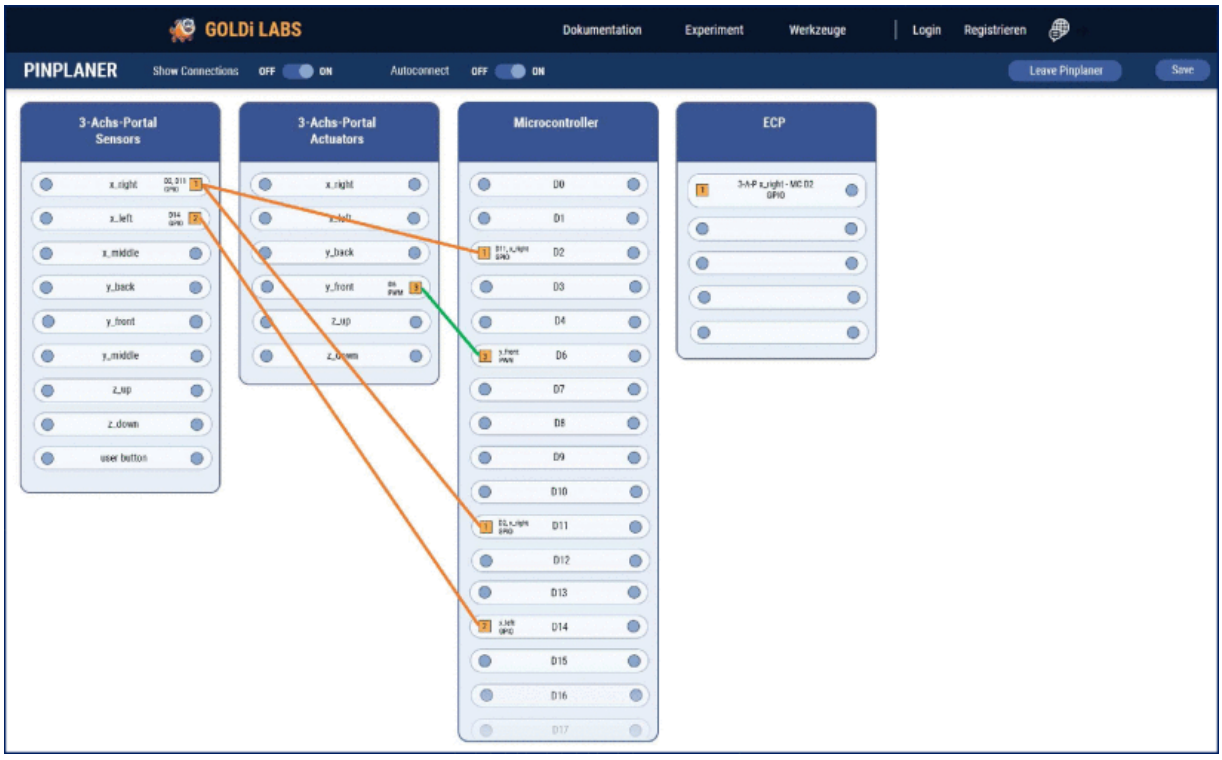


Рисунок 3.5 – Планувальник сигналів/виводів [23]

У випадку Take-Home-Lab перелічено всі контакти роз'єму вводу/виводу, щоб користувач міг пізніше підключити своє обладнання. В ECP всі записи спочатку порожні. Далі маніпулятор миша використовується для вибору частин, що потрібно з'єднати. Наприклад, кінцевий вимикач

датчика *x_right* 3-осьового порталу можна підключити до контакту 1 корпусу Take-Home-Lab, а потім роз'єм 2 до двигуна *engine_right* 3-осьового порталу (див. рис. 3.5). Згідно з цією схемою, вибираються всі сигнальні з'єднання, які будуть використані в експерименті. Для кожного обраного з'єднання автоматично створюється запис всередині ECP, який згодом можна використовувати для спостереження за значеннями сигналів.

Блок 5. Перевірка успішності конфігурування. У разі успішного конфігурування його переносять на обрані лабораторні пристрої та переходять на Блок 6, в іншому випадку – повернення на Блок 2.

Блок 6. Виконання експерименту. Оскільки всі лабораторні пристрої, обрані для цього експерименту, зареєстровані в службі пристроїв GOLDi, під час експерименту запускаються з'єднання WebRTC між пристроями, щоб усі пристрої могли безпосередньо взаємодіяти. Окремий лабораторний сервер більше не потрібен для цієї нової інфраструктури GOLDi 2.0 [23].

Блок 7. Перевірка виконання експеримента. Якщо результати виконання експерименту отримано, тоді на Блок 8, в іншому випадку – повернення на Блок 6.

Блок 8. Кінець.

3.4 Вибір та обґрунтування апаратної платформи розробки

Як пристрій керування може використовуватися МК ATmega2560 встановлений на стороні ВЛ, або МК встановлений на платі Arduino в кейсі Take-Home-Lab (рис. 3.6).

В кейсі Take-Home-Lab встановлено плату від німецької компанії AZ-Delivery, що спеціалізується на доступних мікроконтролерних платах сторонніх виробників, сумісних з Arduino. Ці плати розроблені як сумісні з оригінальним обладнанням Arduino за пінами та кодом, що дозволяє їх програмувати безпосередньо в середовищі розробки Arduino IDE [36]-[37].

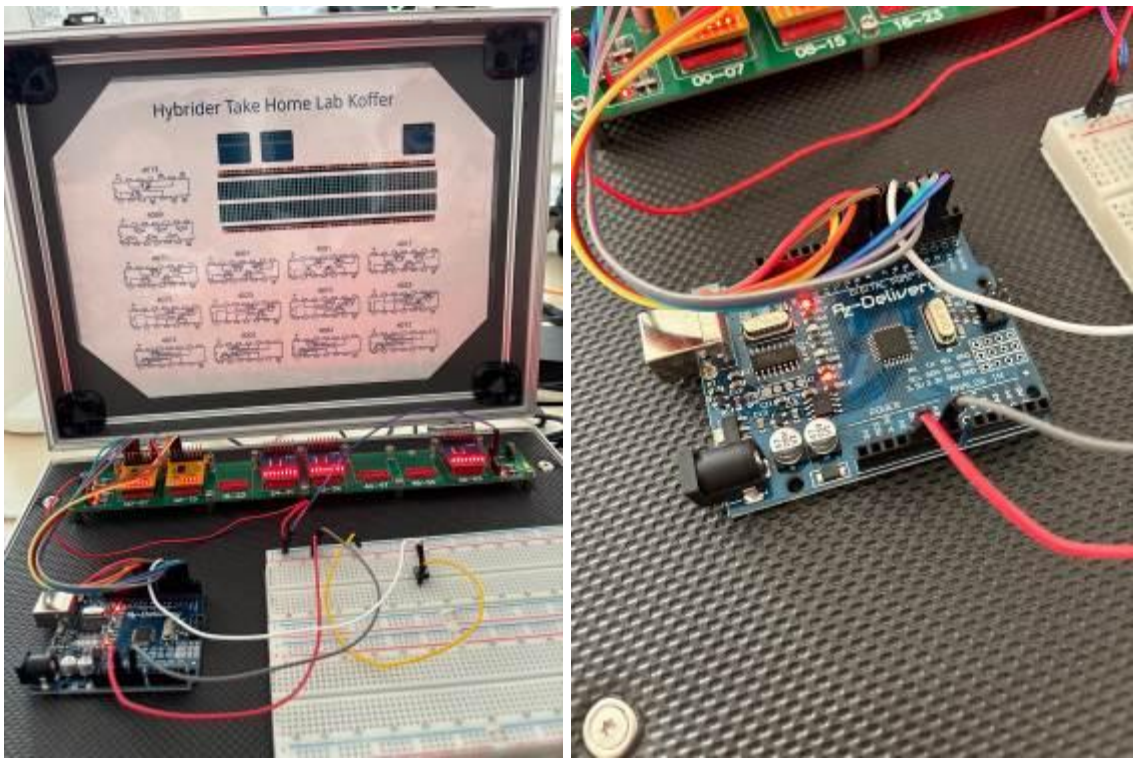


Рисунок 3.6 –Arduino у кейсі Take-Home-Lab

Огляд показав, що популярними рішеннями від компанії AZ-Delivery є такі:

– AZ-Nano V3-Board: невелика плата на базі ATmega328P (16 МГц), зручна для макетування, з 14 цифровими контактами вводу/виводу та 6 аналоговими входами. Ідеально підходить для компактних мобільних проєктів;

– AZ-Delivery UNO R3 (ATmega328P): функціональний клон стандартної Arduino Uno R3, що має той самий чіп ATmega328P, 14 цифрових контактів, 6 аналогових входів та USB-з'єднання для програмування;

– AZ-Mega2560-Board: найбільша та найпотужніша 8-бітна плата на базі ATmega2560. Вона пропонує понад 50 GPIO, більше пам'яті та підходить для більших, складніших проєктів або проєктів з великим обсягом периферійних пристроїв;

– модулі ESP32/ESP8266 (Dev Kit C/NodeMCU): AZ-Delivery пропонує кілька плат із підтримкою Wi-Fi та Bluetooth (наприклад, ESP32-

WROOM-32), сумісних з Arduino IDE, двоядерну обробку (ESP32) для IoT-застосунків [36]-[37].

Результати порівняльного аналізу плат від компанії Arduino та їх аналогів від компанії AZ-Delivery наведено в табл. 3.1 [36]-[38].

Всі плати базуються на 8-біт AVR архітектурі, мають тактову частоту 16МГц та робочу напругу 5 В.

Таблиця 3.1 – Порівняльна характеристика плат від компаній Arduino та AZ-Delivery [36]-[38]

Характеристика	AZ-Nano V3 (CH340)	Arduino Nano V3 (Official)	AZ-Delivery Uno R3	Arduino Uno R3 (Official)	AZ-Mega2560
Мікроконтролер	ATmega328P	ATmega328P	ATmega328P	ATmega328P	ATmega2560
Flash-пам’ять	32 КБ	32 КБ	32 КБ	32 КБ	256 КБ
USB інтерфейс	CH340G (потребує драйверів)	FTDI / ATmega16U2	CH340G (потребує драйверів)	ATmega16U2	CH340G
Цифрові I/O	14 (6 ШІМ)	14 (6 ШІМ)	14 (6 ШІМ)	14 (6 ШІМ)	54 (15 ШІМ)
Аналогові входи	8 (A0-A7)	8 (A0-A7)	6 (A0-A5)	6 (A0-A5)	16
Формфактор	Сумісна з макетною платою	Сумісна з макетною платою	Стандартний шилд	Стандартний шилд	Великий шилд

Ключові відмінності плат від компаній Arduino та AZ-Delivery є такими [36]-[38]:

– USB-чіп (CH340): плати AZ-Delivery часто використовують послідовний перетворювач CH340 замість дорожчих мікросхем FTDI або ATmega16U2, що використовуються на офіційних платах Arduino; тож для розпізнавання плати потрібно встановити драйвер CH340 на ПК;

– завантажувач: деякі плати AZ-Delivery Nano вимагають вибору «ATmega328P (старий завантажувач)» в Arduino IDE для успішного завантаження коду;

– ціна та якість: плати AZ-Delivery зазвичай доступні за нижчою ціною, ніж офіційні плати, часто в комплекті з безкоштовними електронними книгами та навчальними посібниками;

– якість та підтримка: як клони, вони пропонують 100% сумісність для більшості проєктів, але не виробляються офіційною командою Arduino.

Отже, ATmega2560 та ATmega328P – це 8-бітні МК архітектури AVR, головна відмінність між якими полягає в обсязі пам'яті та кількості доступних портів введення-виведення. Відповідно, сферою застосування ATmega328P є прості IoT-пристрої, датчики, компактні гаджети, де важлива низька ціна та мале енергоспоживання; а для ATmega2560 – робототехніка, 3D-принтери, ЧПК-верстати та системи з великою кількістю датчиків та периферії. Порівняльну характеристику цих МК наведено в табл. 3.2 [39]-[41].

Таблиця 3.2 – Порівняльна характеристика МК ATmega2560 та ATmega328P [39]-[41]

Характеристика	ATmega328P	ATmega2560
Flash-пам'ять	32 КБ	256 КБ
ОЗП	2 КБ	8 КБ
EEPROM	1 КБ	4 КБ
GPIO (цифрові виходи)	23 лінії	86 ліній
Аналогові входи (ADC)	6 – 8 каналів	16 каналів
Апаратні UART (Serial)	1 порт	4 порти
Тактова частота	До 20 МГц (зазвичай 16 МГц)	До 16 МГц
Кількість пінів	32 (TQFP) або 28 (DIP)	100 (TQFP)

Ключові відмінності МК такі:

– пам'ять: ATmega2560 має у 8 разів більше Flash-пам'яті та у 4 рази більше оперативної пам'яті, що критично для складних програм з великими бібліотеками або графічними інтерфейсами;

– інтерфейси: завдяки 4 портам UART, ATmega2560 може одночасно працювати з декількома серійними пристроями (наприклад, GPS, Bluetooth та ПК), тоді як ATmega328P обмежений одним портом;

– габарити та складність: ATmega328P доступний у DIP-корпусі, що зручно для макетних плат, тоді як ATmega2560 випускається лише у форм-факторі для поверхневого монтажу (SMT), що складніше для самостійної пайки;

– продуктивність: хоча обидва чіпи працюють на однакових частотах, ATmega2560 може виконувати деякі команди трохи повільніше через необхідність додаткових циклів для доступу до пам'яті вище 64 КБ.

Результати порівняння цих чіпів з точки зору енергоефективності та ресурсоефективності наведені в табл. 3.3 [39]-[41].

Таблиця 3.3 - Порівняння ATmega2560 та ATmega328P з точки зору енергоефективності та ресурсоефективності [39]-[41]

Характеристика	ATmega328P	ATmega2560
Енергоефективність (Power Consumption)		
Струм у режимі сну	0.1 мкА (при 1.8 В)	0.5 – 1 мкА
Струм у активному режимі (на частоті 1 МГц і напрузі 1.8 В)	0.2 мА	0.5 мА
Стабільність на низьких напругах живлення	+	-
Ресурсоефективність (Code & Data Efficiency)		
Адресація пам'яті	Має 16-бітний покажчик команд, що дозволяє звертатися до всієї Flash-пам'яті (32 КБ) миттєво	Пам'ять перевищує ліміт у 64 КБ, тому він використовує 24-бітну адресацію, що змушує процесор витрачати більше тактів на деякі операції переходу та роботу з даними
Оптимізація коду	На малих обсягах даних код для 328P виходить «щільнішим»	Програма може зайняти трохи більше місця через складніші інструкції переходу
Використання ОЗП	Потребує чистого коду без зайвих глобальних змінних	Можливо використовувати «важкі» бібліотеки

Отже, з точки зору енергоефективності лідером є МК ATmega328P, особливо його версія з індексом «P» (Picopower). З точки зору

ресурсоефективності хоча у ATmega2560 більше пам'яті, вона «дорожча» з точки зору обчислювальних витрат. ATmega328P виграє за питомою продуктивністю на кожен витрачений ватт і байт коду. ATmega2560 виграє лише тоді, коли, наприклад, потрібен великий буфер для дисплея або масив даних.

3.5 Вибір технологій програмної реалізації та середовища розробки

Розробка програми може здійснюватися в середовищі Arduino IDE (рис. 3.7). Крім того, для редагування вихідного коду в браузері та його компіляції доступне вебінтегроване середовище проєктування WIDE (рис. 3.8), доступ до якого можна отримати через вебінтерфейс онлайн-лабораторії GOLDi, що забезпечує єдиний інтерфейс користувача для всіх мов програмування [28].

Програмування для Arduino базується на мові C++, що дозволяє використовувати як структурний (процедурний), так і об'єктно-орієнтований підхід (ООП). Вибір між ними залежить від складності проєкту: структурний підхід кращий для простих завдань, тоді як ООП спрощує управління великими та складними системами [42]-[44].

Структурне (процедурне) програмування – це класичний підхід для Arduino, де код розбивається на послідовність дій та функцій. Він фокусується на логічній структурі та процесі виконання. Його основою є функції налаштування setup() та основного циклу loop().

Як основні особливості можна відмітити такі: простіший для розуміння початківцями; використовує процедури (функції) для організації коду; дані (змінні) часто є глобальними [42]-[44].

Рисунок 3.7 – Приклад коду в Arduino IDE

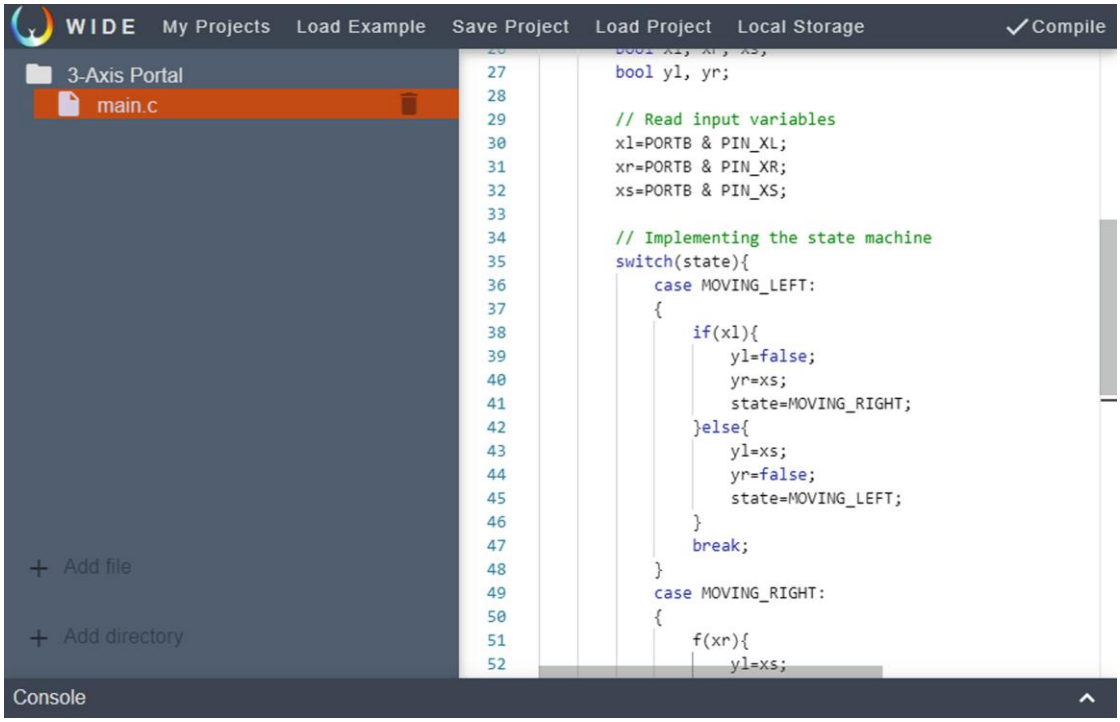


Рисунок 3.8 – Приклад коду в середовищі програмування WIDE [28]

В якості прикладу можна навести код такий код.

```
int ledPin = 13;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

ООП базується на створенні «об'єктів», які об'єднують дані (властивості) та методи (функції) для роботи з цими даними. Це дозволяє уникнути дублювання коду та робити проєкти більш масштабованими. Його основою є класи, об'єкти, інкапсуляція, успадкування [42]-[44].

Як основні особливості можна відмітити такі:

- спрощує роботу з багатьма компонентами (наприклад, 10 світлодіодів);
- краще структурує складні проєкти (наприклад, роботу з дисплеєм, датчиками та меню);
- більш інтенсивно використовує пам'ять, хоча для сучасних Arduino це зазвичай не є проблемою.

В якості прикладу можна навести такий код.

```
class Led {
  byte pin;
  public:
    Led(byte p) : pin(p) {} // Конструктор
    void begin() { pinMode(pin, OUTPUT); }
    void on() { digitalWrite(pin, HIGH); }
    void off() { digitalWrite(pin, LOW); }
```

```
};
Led myLed(13); // Створення об'єкта
void setup() { myLed.begin(); }
void loop() {
  myLed.on(); delay(1000);
  myLed.off(); delay(1000);
}
```

Результати порівняння технологій розробки наведено в табл. 3.4 [42]-[44].

Таблиця 3.4 – Порівняння технологій розробки [42]-[44]

Характеристика	Структурне програмування	Об'єктно-орієнтоване програмування
Фокус	Процедури/функції	Дані/об'єкти
Підхід	Top-down (зверху-вниз)	Bottom-up (знизу-вгору)
Складність	Легкий для невеликих проєктів	Потребує більше часу на проєктування
Масштабованість	Важко підтримувати великий код	Легко розширювати
Модифікованість	Складніше модифікувати програми та повторно використовувати код	Менш складно модифікувати програми та повторно використовувати код
Гнучкість	Забезпечує меншу гнучкість та абстракцію	Забезпечує більшу гнучкість та абстракцію
Приклад в Arduino	digitalWrite(), delay()	Servo myservo, Serial.print()

В якості середовища розробки для Arduino можливо використовувати різні інструменти, які спрощують написання програм на C/C++, компіляцію та завантаження скетчів через USB, що робить їх зручними як для початківців, так і для досвідчених розробників [45]-[46].

Офіційними інструментами від компанії Arduino є такі:

– Arduino IDE 2.x – це сучасна версія класичного середовища, що має автодоповнення коду, вбудований налагоджувач (debugger), інтеграцію з git;

– Arduino Cloud Editor – це онлайн-редактор, який дозволяє писати та завантажувати код прямо з браузера, зберігаючи проєкти в хмарі;

– Arduino IDE 1.8.x (Legacy) – це класична «легка» версія, яку досі використовують для старих систем або слабких ПК.

Для складних проєктів можливо використувати такі середовища розробки:

– VS Code + PlatformIO – це найпопулярніший вибір для професіоналів, що підтримує тисячі плат та платформ, має потужний менеджер бібліотек, інтелектуальне доповнення коду, забезпечує швидшу розробку, зручну роботу з великими проєктами;

– Visual Studio + Visual Micro – це розширення для повноцінної Visual Studio, що додає професійні інструменти налагодження та аналізу коду;

– CLion + PlatformIO – це IDE від JetBrains для розробників, що звикли до преміальних інструментів розробки на C++.

Результати порівняльного аналізу середовищ розробки для Arduino наведено в табл. 3.5 [45]-[46].

Отже, найкращі альтернативи Arduino IDE пропонують розширені функції, такі як покращене автодоповнення коду, налагодження та управління проєктами. В той же час середовище Arduino IDE 2.3.7 також надає функції автодоповнення коду, вбудований дебагер, простий інтерфейс, тому є гарним вибором для студентів.

3.6 Моделювання процесів опрацювання даних при взаємодії з віддаленою лабораторією

Для опрацювання даних в рамках представленої архітектури (див. рис. 3.1) було розроблено дві моделі, що відповідно описують процеси взаємодії з ВЛ на основі апаратно-орієнтованого та віртуалізованого підходів.

Таблиця 3.5 – Порівняльна характеристика середовищ розробки для Arduino [45]-[46]

Середовище розробки	Рівень досвіду	Ключові переваги	Основні недоліки
Arduino IDE 2.x	Початковий/ Середній	Автодоповнення коду, вбудований дебагер, простий інтерфейс	Менша гнучкість для професійних проєктів
Arduino Cloud/Web Editor	Початковий	Робота в браузері, доступ до коду з будь-якого пристрою, синхронізація	Потрібен стабільний Інтернет, обмежена підтримка сторонніх бібліотек
VS Code + PlatformIO	Середній/ Професійний	Розумний менеджер бібліотек, підтримка великої кількості платформ та плат, автодоповнення коду, вбудований дебагер	Складніше налаштування. Вимагає часу на вивчення конфігурації
Arduino IDE 1.8.x	Початковий	Стабільність, низькі вимоги до ПК, величезна база прикладів	Застарілий інтерфейс, відсутність автодоповнення та дебагу
CLion + PlatformIO	Професійний	Найкращий статичний аналіз коду та рефакторинг, професійні інструменти JetBrains	Платна ліцензія (для CLion), високі вимоги до ресурсів ПК

Модель взаємодії з ВЛ на основі апаратно-орієнтованого підходу реалізована у вигляді діаграми станів скінченного автомата Мілі (рис. 3.9), де переходи між станами ініціюються комбінацією поточного стану та зміною вхідних сигналів від сенсорів.

Після подачі живлення або апаратного скидання (Reset) система переходить у стан ініціалізації Homing, виконання команд у межах якого забезпечує гарантоване повернення каретки портала у стартову позицію. Успішне спрацьовування відповідних кінцевих вимикачів переводить автомат у стан очікування Wait, що відображає інтерактивний підхід. У цьому стані виконання призупиняється до моменту реєстрації зовнішнього тригера – натискання фізичної кнопки користувачем. Після активації викликається блок прийняття рішень щодо маршруту choosePath(), який відповідає за рандомізацію траєкторії та розподіляє виконання на один із

шести можливих станів послідовного лінійного або діагонального переміщення, що залежить від поточного напрямку руху та випадковості.

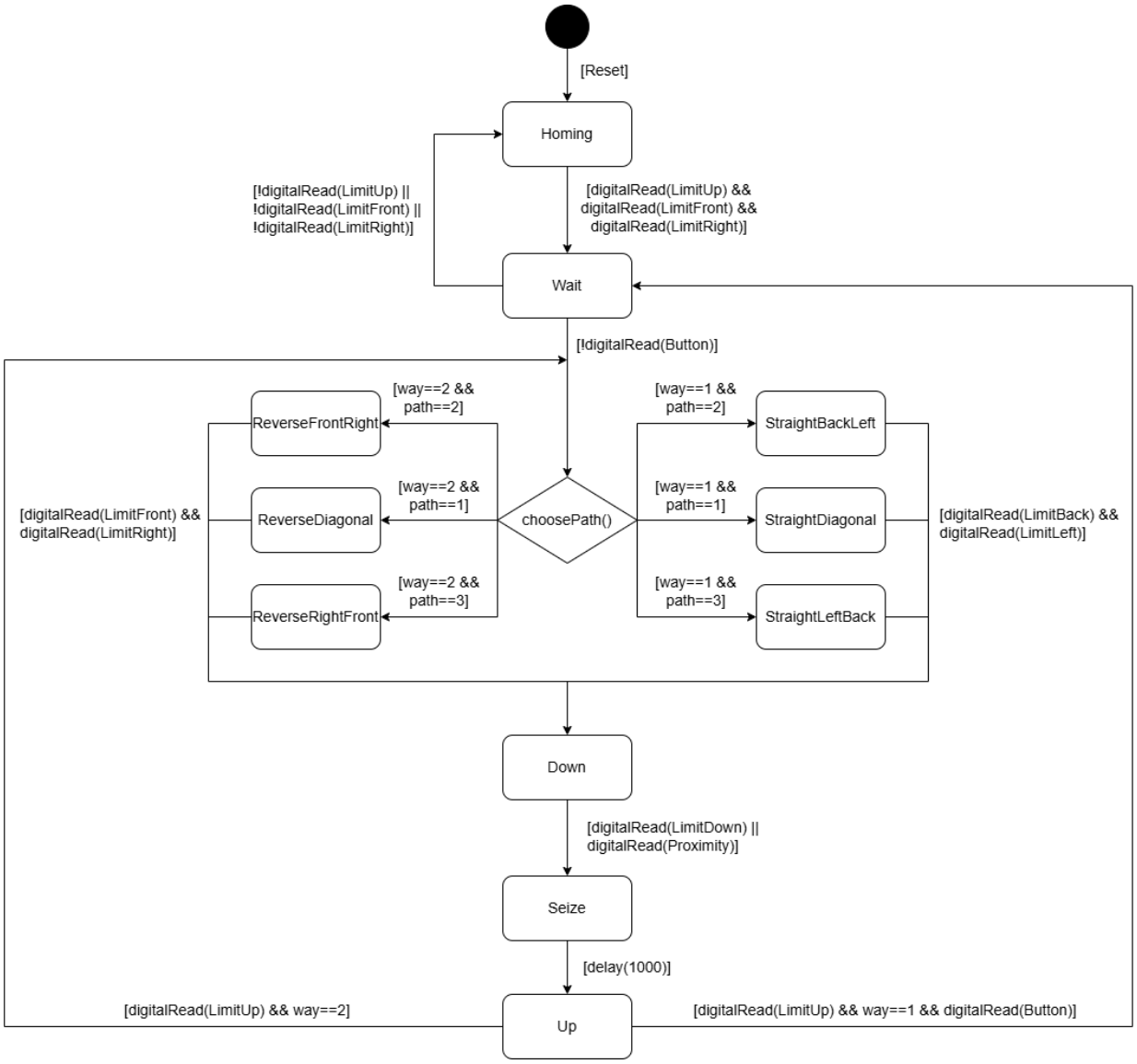


Рисунок 3.9 – Модель взаємодії з ВЛ на основі апаратно-орієнтованого підходу у вигляді діаграми станів скінченного автомата Мілі

Досягнувши цільових координат, автомат послідовно проходить стадії опускання захоплювача (Down), вмикання (або вимикання) електромагніту для фіксації (або звільнення) вантажу (Seize) із регламентованою часовою затримкою в одну секунду, а також підняття (Up). Залежно від напрямку руху, після підняття система або повертається до блоку choosePath() для генерації маршруту для повернення, або переходить назад у стан Wait для

очікування наступної команди користувача. Крім того, передбачено захисний зворотний зв'язок: якщо під час очікування каретка зміститься з базової точки, система автоматично ініціює повторний цикл Homing.

Модель взаємодії з ВЛ на основі віртуалізованого підходу також представлена у вигляді діаграми станів скінченного автомата Мілі (рис. 3.10), проте має суттєві структурні відмінності, зумовлені парадигмою виконання без фізичного втручання.

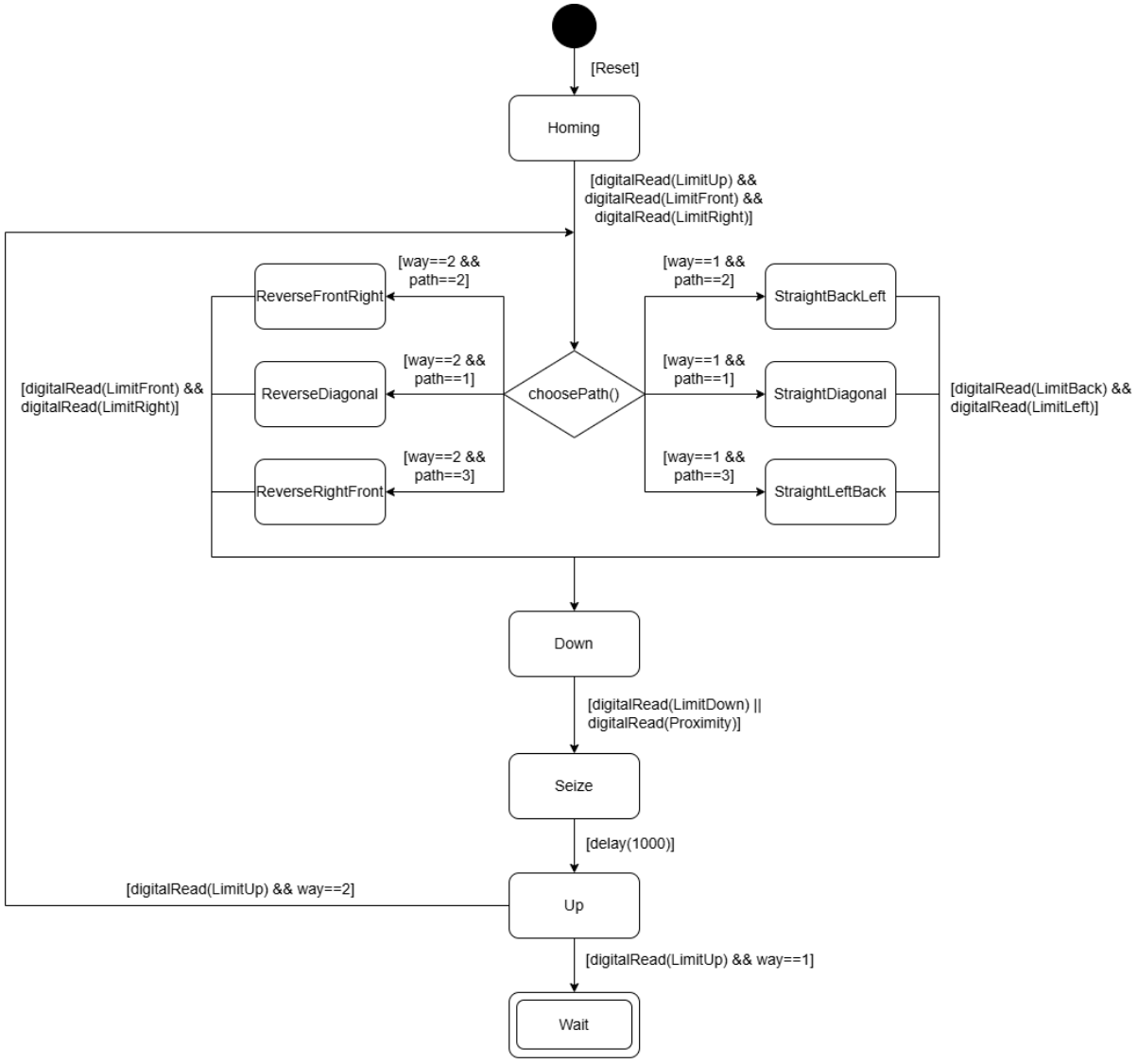


Рисунок 3.10 – Модель взаємодії з ВЛ на основі віртуалізованого підходу у вигляді діаграми станів скінченного автомата Мілі

Після ініціалізації та успішного завершення фази позиціювання Homing, автомат не переходить в режим очікування дій користувача, а безперервно продовжує виконання програми, одразу переходячи до блоку генерації маршруту choosePath(). Процес переміщення, опускання, перемикання режиму магніту та наступного підняття захоплювача відбувається ідентично до першої моделі. Головна концептуальна відмінність полягає у фінальній стадії життєвого циклу програми. Після успішного завершення повного циклу руху та повернення у вихідну точку, система переходить у кінцевий стан Wait. Оскільки в цій конфігурації відсутні зовнішні елементи інтерактивного керування, такий стан є безвихідним для програмної логіки. МК залишається у ньому нескінченно довго, запобігаючи неконтрольованим повторним запускам механізмів. Вихід із цього стану можливий виключно через отримання глобального сигналу апаратного скидання (Reset) від серверного шлюзу. Після цього можливо ініціалізувати повторне виконання експерименту.

3.7 Висновки до розділу 3

В цьому розділі виконано аналіз вимог до розробки, наведено алгоритм програмно-апаратної взаємодії з віддаленим експериментом за допомогою кейсу Take-Home-Lab. Обґрунтовано вибір апаратних засобів та технологій розробки програми для опрацювання даних експерименту у ВЛ на основі використання МК ATmega2560 і ATmega328P та середовища Arduino IDE 2.3.7. Розроблено моделі опрацювання даних при взаємодії з ВЛ.

4 РОЗРОБКА ТА ПРАКТИЧНЕ ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОПРАЦЮВАННЯ ДАНИХ У ВІДДАЛЕНІЙ ЛАБОРАТОРІЇ

4.1 Опис віддаленого експерименту

Модель 3-осьового порталу імітує стаціонарного робота-маніпулятора з ортогональним робочим простором, який використовується для передачі заготовок до обробного або сортувального блоку, що використовується, наприклад, на автоматизованому виробництві (рис. 4.1). Модель складається з каретки, здатної рухатися в двох горизонтальних лінійних напрямках, та електромагнітного захоплювача, придатного для переміщення у вертикальному напрямку Z. Базовий портал додатково доповнюється набором деталей, що здатні до намагнічування, а також станціями завантаження та розвантаження. Кінцеві положення кількох рухомих частин моделі фіксуються апаратними кінцевими вимикачами, а наявність об'єкта під захоплювачем додатково може контролюватися датчиком наближення. Переміщення за осями X та Y можливе лише у випадку, коли захоплювач знаходиться у верхньому кінцевому положенні. Кінцеві вимикачі допомагають запобігти пошкодженню порталу внаслідок виходу за межі дозволеного робочого простору через помилку в програмуванні блоку керування – у разі спрацьовування вони призводять до негайної зупинки відповідної осі.

На основі цього експерименту можна виконати моделювання такого виробничого процесу: металева заготовка піднімається електромагнітним захоплювачем з визначеної точки завантаження, переміщується до точки розвантаження та там опускається, після чого робот може повернутись за новою заготовкою. На початку роботи алгоритму рухомі частини портального крана здійснюють процедуру повернення у базову стартову позицію. На відміну від систем з інкрементальними енкодерами вимірювання відстані, у цій моделі позиціювання спирається на дискретні сигнали:

захоплювач рухається спочатку вгору, а далі в горизонтальній площині у визначений базовий кут робочого простору, доки не спрацюють відповідні кінцеві вимикачі.

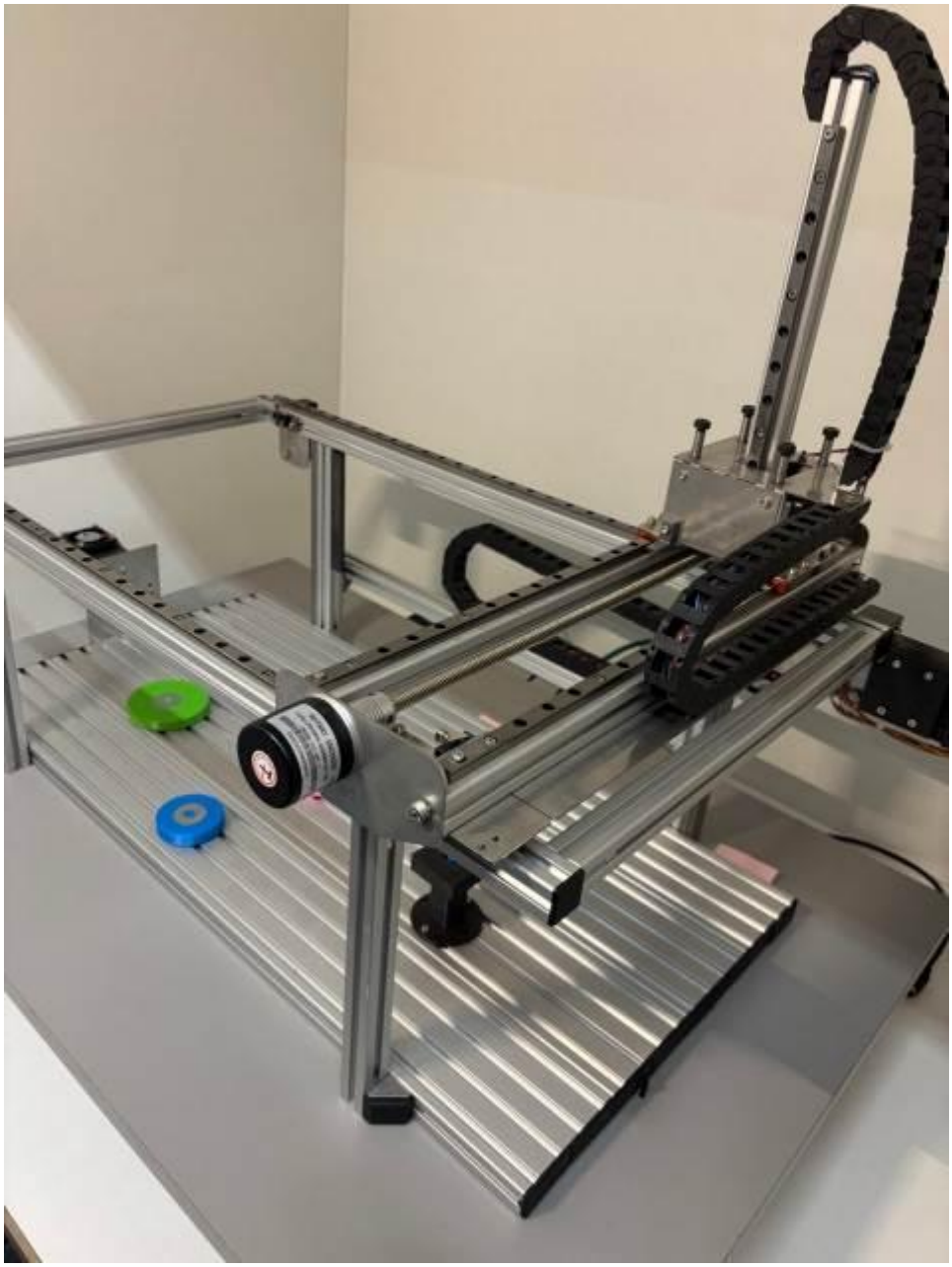


Рисунок 4.1 – Вигляд фізичного обладнання експерименту

Після успішної просторової ініціалізації запускається основний цикл переміщення. Система здатна реалізовувати різні траєкторії руху для досягнення цільової зони в межах робочого простору (пряма діагональна траєкторія або L-подібне послідовне переміщення за осями X та Y), що надає

можливість програмної рандомізації шляху. Згідно з обраним шляхом, каретка переміщується в напрямку X та Y, доки не досягне потрібного положення над заготовкою. Далі відбувається рух в напрямку -Z, доки датчик наближення не просигналізує про те, що захоплювач торкнувся об'єкта. Магніт вмикається й деталь кріпиться до захоплювача. Після певної затримки захоплювач рухається в напрямку +Z, доки знову не досягне свого верхнього кінцевого положення. Після цього, каретка знову виконує рух у горизонтальній площині, змінюючи координати X та Y, доки не досягне позиції розвантаження. Досягнувши цього стану, захоплювач знову рухається в напрямку -Z, доки прикріплена деталь не буде розміщена на поверхні, де її, як і раніше, розпізнає індуктивний датчик наближення. Заготовка вивантажується шляхом деактивації магніту, і захоплювач піднімається в напрямку +Z [47].

Схему експерименту наведено на рис. 4.2. В табл. 4.1 наведено опис та відповідність пінів Arduino UNO та ATmega2560 до пінів сенсорів, а в табл. 4.2 – опис та відповідність пінів Arduino UNO та ATmega2560 до пінів актуаторів.

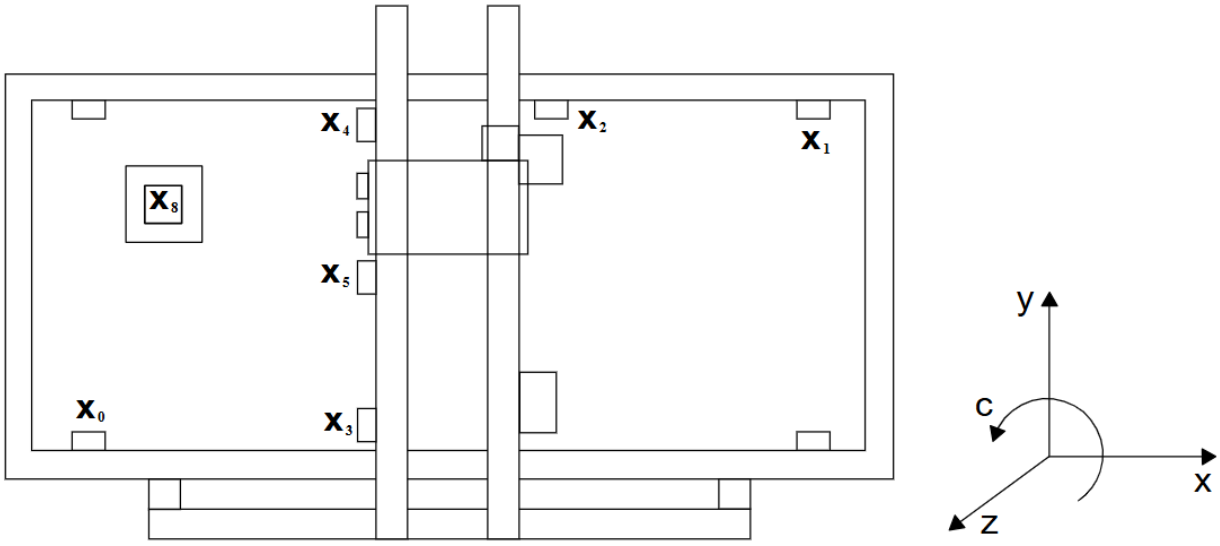


Рисунок 4.2 – Схема експерименту [47]

Таблиця 4.1 – Опис та відповідність пінів Arduino UNO та ATmega2560 до пінів сенсорів

Піни Arduino UNO до пінів сенсорів	Піни ATmega2560 до пінів сенсорів	Опис	Тип
2 – D00	3 – PE5	LimitLeft	Input
3 – D01	2 – PE4	LimitRight	Input
4 – D02	4 – PG5	LimitBack	Input
5 – D03	5 – PE3	LimitFront	Input
6 – D04	13 – PB7	LimitDown	Input
7 – D05	14 – PJ1	LimitUp	Input
8 – D06	15 – PJ0	Proximity	Input
A3	–	Button	User Input

Таблиця 4.2 – Опис та відповідність пінів Arduino UNO та ATmega2560 до пінів актуаторів

Піни Arduino UNO до пінів актуаторів	Піни ATmega2560 до пінів актуаторів	Опис	Тип
9 – D07	8 – PH5	DriveLeft	Output
10 – D08	7 – PH4	DriveRight	Output
11 – D09	9 – PH6	DriveBack	Output
12 – D10	10 – PB4	DriveFront	Output
A0 – D11	11 – PB5	DriveDown	Output
A1 – D12	12 – PB6	DriveUp	Output
A2 – D13	30 – PC7	Magnet	Output

Логіку просторових переміщень формалізовано у вигляді розроблених моделей взаємодії – діаграм станів скінченного автомата Мілі (див. рис. 3.9-3.10). Залежно від обраного підходу взаємодії з ВЛ, система після завершення повного циклу маніпуляцій переходить у стан очікування, де діє за різними сценаріями – очікує або команду від користувача, або апаратне перезавантаження. У кожному зі станів автомата керуючі впливи на

електродвигуни осей формуються динамічно, базуючись на поточному стані системи та миттєвих значеннях сигналів від кінцевих вимикачів.

4.2 Структура програми та опис особливостей програмної реалізації

Узагальнену структуру програми можна представити у вигляді набору модулів (рис. 4.3), що пов'язані між собою та є спільними для обох версій програмної реалізації представлених моделей взаємодії (див. підрозділ 3.6).

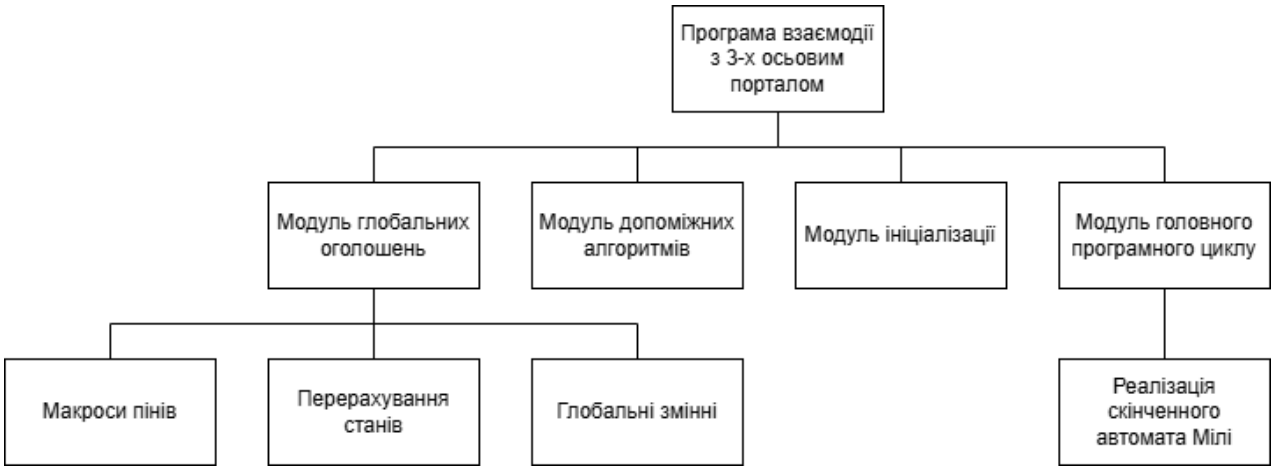


Рисунок 4.3 – Узагальнена структура програми

Структура обох програм складається з чотирьох основних модулів:

– модуль глобальних оголошень;
– модуль допоміжних алгоритмів;
– модуль ініціалізації;
– модуль головного програмного циклу.

Реалізацію програм було виконано у середовищі розробки Arduino IDE 2.3.7 засобами мови C++ з використанням фреймворку Arduino Core.

Враховуючи архітектурні особливості та відмінності (див. рис. 3.1), для двох підходів до взаємодії з ВЛ (на базі апаратно-орієнтованого та віртуалізованого підходів) було створено два окремі файли вихідного коду:

case_code_v1.ino (Додаток А.1) та standalone_code_v1.ino (Додаток А.2) відповідно.

Модуль глобальних оголошень відповідає за препроцесорну обробку та ряд декларацій: тут визначено макроси для апаратних пінів сенсорів та актуаторів згідно з розробленими таблицями відповідності (див. табл. 4.1-4.2), користувацький тип даних States_t для переліку всіх можливих станів системи (див. рис. 3.9-3.10) та глобальні змінні.

Модуль допоміжних алгоритмів містить функцію choosePath, що реалізує математичну логіку генерації псевдовипадкових чисел для вибору траєкторії руху каретки та забезпечує подальший перехід автомата до обраного стану.

Модуль ініціалізації, реалізований у вигляді функції setup, виконується одноразово при старті системи: він конфігурує напрямки роботи портів вводу/виводу; підключає внутрішні підтягувальні резистори (INPUT_PULLUP) для захисту від хибних спрацьовувань; забезпечує, що початкове значення напруги на пінах дорівнює 0; переводить автомат у початковий стан Homing та визначає прямий напрям руху портального крану.

Модуль головного програмного циклу, що реалізований у вигляді функції loop, містить безпосередню реалізацію логіки скінченного автомата. Тут описані команди, що мають бути виконані в межах кожного з 11 можливих станів: Wait, StraightBackLeft, StraightLeftBack, StraightDiagonal, ReverseFrontRight, ReverseRightFront, ReverseDiagonal, Down, Up, Seize та Homing; а також умови переходів між ними.

Основні відмінності програми standalone_code_v1.ino від програми case_code_v1.ino полягають у такому:

– відсутність макросу піна фізичної кнопки та будь-яких перевірок чи зчитувань, пов'язаних з нею;

– змінена логіка кінцевого стану Wait;

– змінена логіка переходу з початкового стану Homing.

Крім того, відрізняється фінальний етап обробки розробленого коду середовищем Arduino IDE. Для Arduino UNO відбувається пряма прошивка через USB кабель, а для ATmega2560 виконується експорт скомпільованого бінарного файлу у форматі *.hex для подальшої передачі через віртуальний інтерфейс лабораторії.

Однією з особливостей програмної реалізації є те, що для оптимізації використання оперативної пам'яті МК усі апаратні адреси пінів були задані не як змінні, а як директиви препроцесора #define. Це дозволяє компілятору підставляти безпосередні константи замість рядків вихідного коду у рамках першого етапу компіляції. У якості ідентифікаторів використані назви, що виступають як опис у табл. 4.1 та табл. 4.2; а значеннями стали власне номери пінів Arduino UNO та ATmega2560 з аналогічних таблиць. Усі локальні та глобальні змінні, що беруть участь у збереженні проміжних значень під час виконання алгоритмів та не були замінені макросами, наведено у табл. 4.3. Для забезпечення взаємодії з апаратною частиною та реалізації логіки програми було використано набір стандартних функцій фреймворку Arduino Core, а також власну розроблену функцію рандомізації. Їх опис наведено у табл. 4.4.

Таблиця 4.3 – Опис користувацьких змінних

Назва змінної	Тип змінної	Опис
state	States_t (користувацький перелік на основі enum)	Глобальна змінна, що зберігає поточний активний стан скінченного автомата
way	int (стандартний цілочисельний)	Глобальна змінна, що відображає напрямок руху в межах поточного циклу. Значення 1 показує прямий рух (до заготовки), а 2 – зворотний рух (до початкової точки)
path	int (стандартний цілочисельний)	Локальна змінна всередині функції choosePath, що зберігає згенероване псевдовипадкове число від 1 до 3 для вибору маршруту

Таблиця 4.4 – Опис використаних функцій

Функція	Опис
Користувацька	
choosePath()	Функція рандомізації без вхідних параметрів або значення, що повертається (void). Генерує випадковий маршрут переміщення крана. Використовує змінні *way*, *path* та *state*
Arduino Core	
setup()	Головна функція ініціалізації МК без вхідних параметрів або значення, що повертається (void). Викликається один раз після апаратного скидання. Встановлює початкові значення *state* та *way* (відповідно Homing та 1)
loop()	Головна функція нескінченного циклу без вхідних параметрів або значення, що повертається (void). Реалізує основну логіку роботи програми – зміну станів автомата Мілі за допомогою switch-case. Змінює та зчитує значення *state* та *way*
pinMode(*X*, *Y*)	Встановлює для заданого піна *X* режим роботи *Y*: як вхід INPUT, вихід OUTPUT або вхід з підтягувальними резисторами INPUT_PULLUP
digitalWrite(*X*, *Y*)	Встановлює логічний рівень *Y* (HIGH або LOW) на заданому піні *X* для керування актуаторами (моторами або електромагнітом)
digitalRead(*X*)	Зчитує поточний логічний стан із піна сенсора *X* (кінцевого вимикача, датчика наближення або кнопки). Повертає значення 1 або 0
delay(*C*)	Призупиняє виконання програми на *C* мілісекунд
micros()	Повертає кількість мікросекунд з моменту запуску плати
randomSeed(*C*)	Ініціалізує генератор псевдовипадкових чисел за допомогою значення початкової точки *C* (в якості цього значення було використано результат функції micros, що забезпечує повну випадковість під час кожного запуску)
random(*A*, *B*)	Генерує псевдовипадкове ціле число у заданому діапазоні від *A* (включно) до *B* (не включно). За допомогою цієї функції реалізовано рандомізацію у choosePath

4.3 Практична реалізація експериментів та функціональне тестування програм

Для перевірки коректності розроблених алгоритмів та підтвердження працездатності програмного коду було практично відтворено експеримент та проведено відповідне функціональне тестування на базі реального електромеханічного обладнання ВЛ GOLDi. Процес реалізації експерименту відбувався через вебпортал системи у чітко визначеній послідовності.

Першим кроком є авторизація користувача на вебсайті GOLDi LABS (рис. 4.4).

Рисунок 4.4 – Вебпортал GOLDi LABS

Після успішного входу, необхідно провести налаштування експерименту (рис. 4.5): обрати тип обладнання, що буде використано (наприклад, 3-х осьовий портал), а також модуль для керування цим обладнанням (microcontroller_01 або кейс MOLE 20).

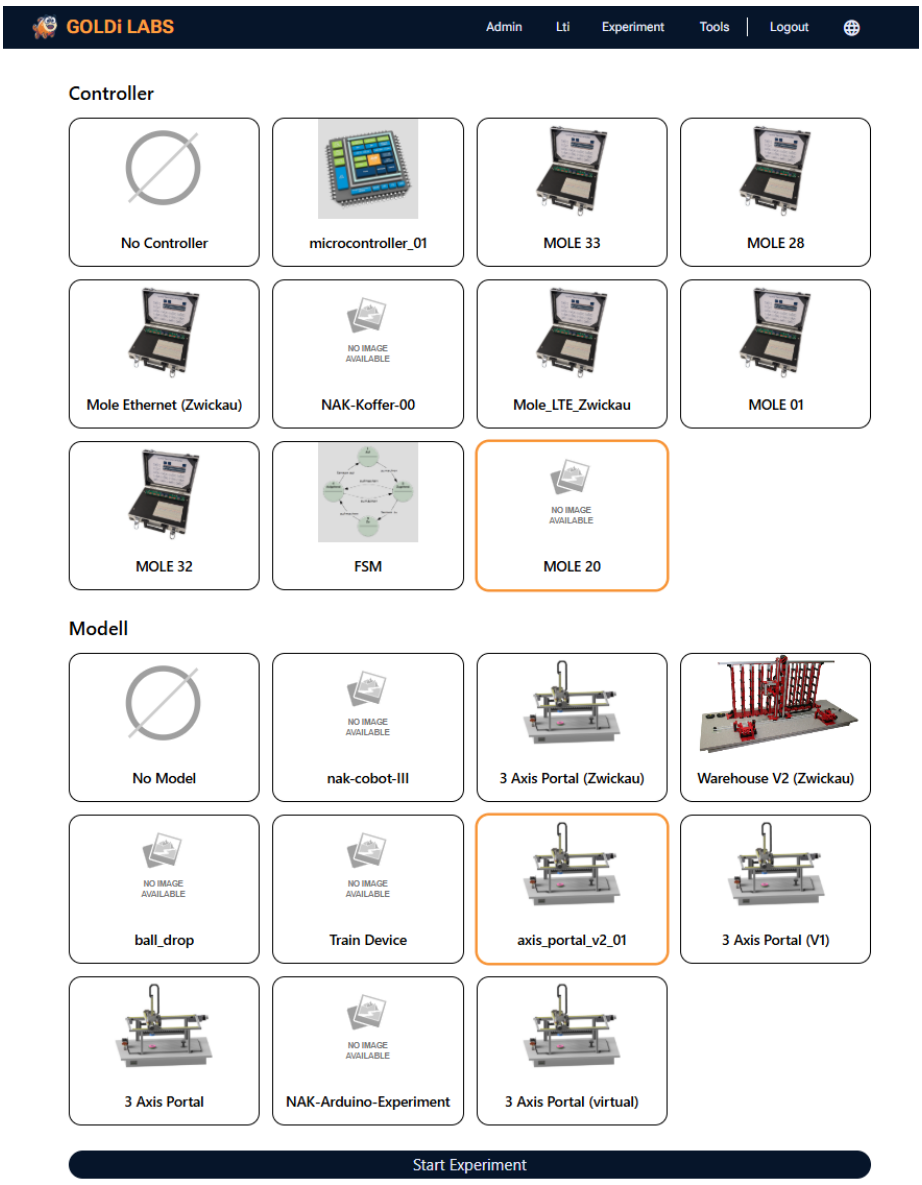


Рисунок 4.5 – Налаштування експерименту

Після визначення цих параметрів, необхідно перевірити статус підключення обраної апаратної частини до хмарної інфраструктури (рис. 4.6). У випадку успішної верифікації можна переходити до завершального етапу налаштування – генерації конфігураційного файлу експерименту в JSON форматі (рис. 4.7).

Цей файл описує всі системні налаштування: розподіл ролей між мережевими вузлами; шляхи до девайсів у системі; визначення інтерфейсів; жорстку відповідність цифрових пінів контролера до фізичних сенсорів та актуаторів лабораторного стенда.

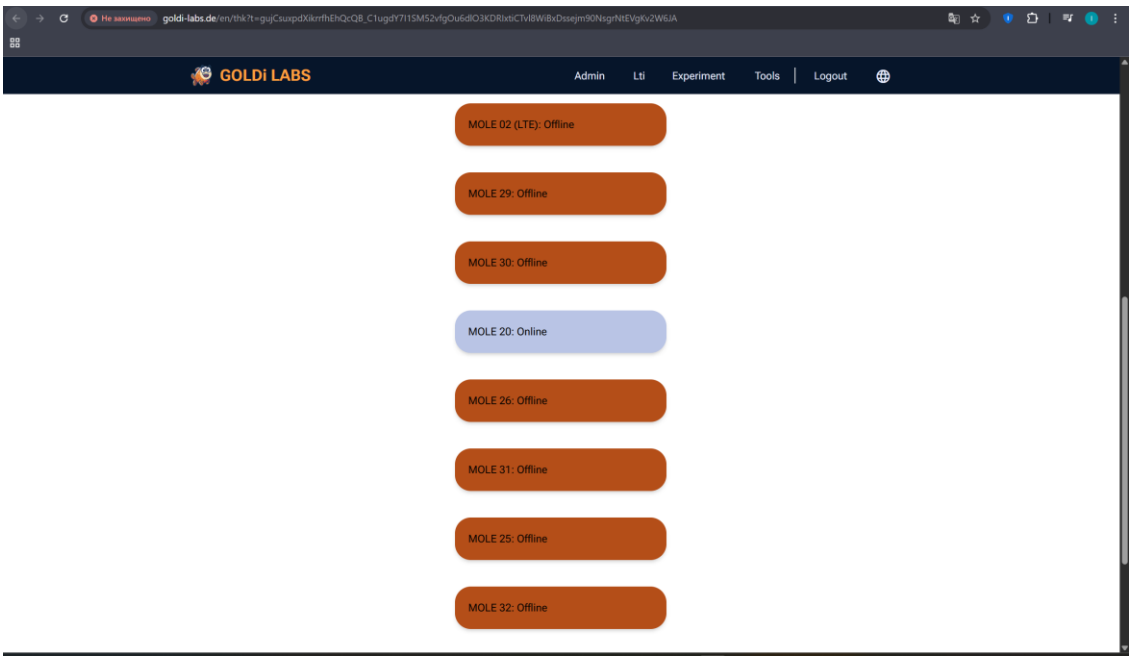


Рисунок 4.6 – Перевірка статусу підключення до хмарної інфраструктури

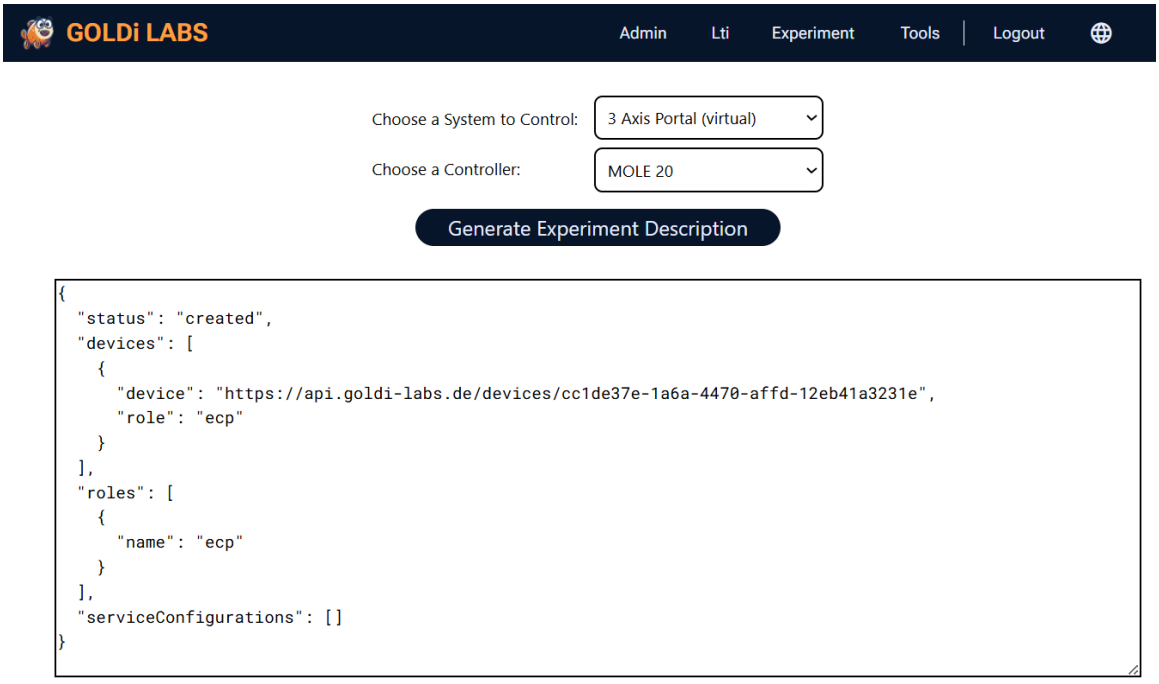


Рисунок 4.7 – Генерація конфігураційного файлу експерименту

Саме цей згенерований конфігураційний файл забезпечує програмно-апаратну сумісність між розробленим кодом та віддаленим обладнанням.

Наступним кроком є безпосередній запуск експериментальної сесії та завантаження ПЗ (рис. 4.8).

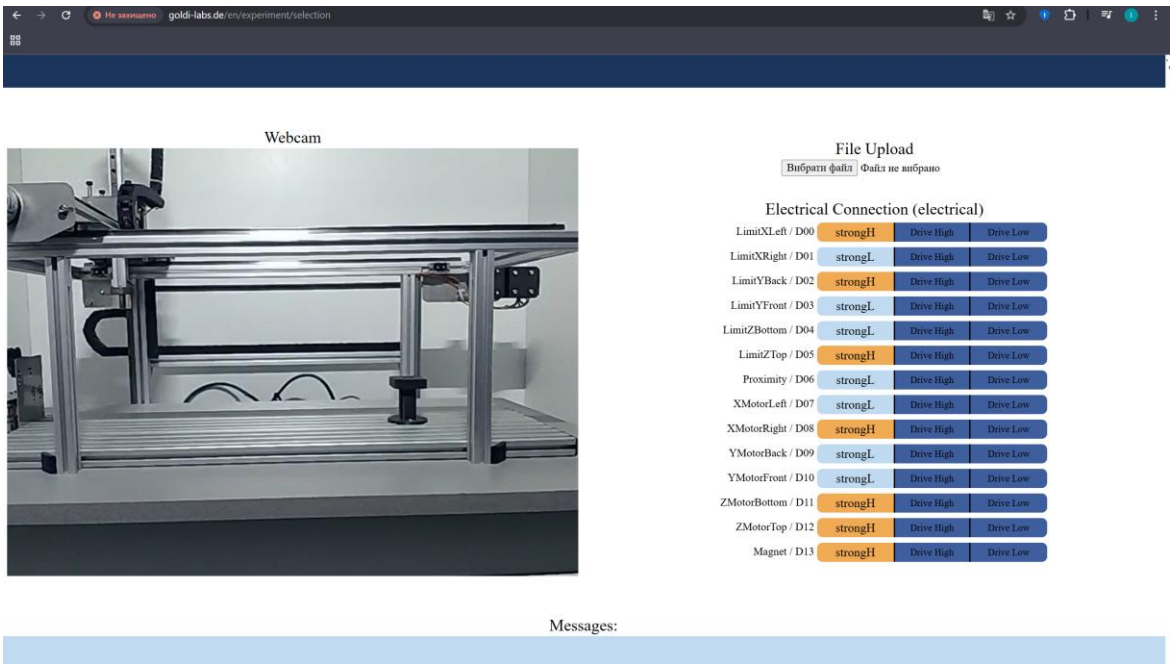


Рисунок 4.8 – Активна експериментальна сесія

Процедура ініціалізації відрізняється залежно від обраного підходу до взаємодії з ВЛ. У випадку використання вбудованого контролера ATmega2560, після активації експерименту необхідно завантажити попередньо скомпільований бінарний файл формату *.hex (інтерфейс надає відповідну кнопку завантаження після запуску). Виконання алгоритму розпочинається після того, як система запрограмує та перезавантажить МК. У сценарії з використанням апаратного кейса, необхідно, щоб Arduino UNO вже мала завантажений код ще до початку сесії. Тоді після запуску експерименту в браузері система одразу перейде до виконання команд.

В ході проведення серії тестувань було встановлено, що обидві моделі взаємодії функціонують коректно та у повній відповідності до розроблених діаграм станів скінченних автоматів Мілі. Актуатори коректно та без затримок виконували команди; сенсори своєчасно фіксували кінцеві положення; стани змінювались у коректному порядку, що забезпечило успішне переміщення заготовки магнітом; при цьому відеопотік дозволяв візуально контролювати процес у реальному часі. Для систематизації результатів перевірки головних функціональних елементів кожного

алгоритму було складено план тестування, результати якого наведено у табл. 4.5.

Таблиця 4.5 – Результати функціонального тестування програм взаємодії з ВЛ

Елемент, що тестується	Очікуваний результат	Статус виконання
Процедура ініціалізації (Homing)	Кран коректно переміщується у початковий кут (вгору, вперед, вправо) до почергового спрацьовування відповідних кінцевих вимикачів	Успішно
Рандомізація траєкторії руху	При кожному новому циклі функція choosePath() генерує новий маршрут; кран візуально змінює схему руху (діагонально або L-подібно)	Успішно
Робота датчика наближення	При опусканні захоплювача рух вниз (-Z) миттєво зупиняється у момент виявлення металевої заготовки індуктивним датчиком	Успішно
Керування електромагнітом	Магніт активується після досягнення заготовки, надійно утримує її під час переміщення та деактивується у точці розвантаження із заданою затримкою в 1 секунду	Успішно
Послідовність станів (реалізація автомату Мілі)	Алгоритми чітко дотримуються логіки, описаної у моделях взаємодії: переходи виконуються коректно, стани не пропускаються, неконтрольованй рух за осями відсутній	Успішно
Інтерактивне керування кнопкою (взаємодія через кейс)	Після завершення циклу, система зупиняється у стані Wait. Повторний рух починається виключно після натискання фізичної кнопки користувачем	Успішно
Реакція на апаратне скидання	Надсилання сигналу Reset для перезавантаження або завершення експерименту миттєво перериває будь-який поточний рух. Після перезапуску початковим станом є Homing	Успішно

Враховуючи отримані результати, можна зробити висновок, що розроблене ПЗ повністю задовольняє поставлені вимоги та забезпечує стабільне виконання поставлених завдань.

4.4 Порівняльна характеристика способів взаємодії з віддаленою лабораторією

Проведені експерименти дозволили практично реалізувати та дослідити два концептуально різні підходи щодо взаємодії з ВЛ: апаратно-орієнтований та віртуалізований. Щоб визначити оптимальну стратегію використання кожного з них, було проведено порівняльний аналіз (табл. 4.6). Оцінювання здійснювалося за критеріями гнучкості та масштабованості, результатів навчання, сучасних вимог до ресурсо- та енергоефективності зелених інформаційних технологій (ІТ) та ін. [48]- [51].

Таблиця 4.6 – Порівняльна характеристика способів взаємодії з ВЛ

Критерій оцінювання	Апаратно-орієнтований (кейс Hybrid Take-Home-Lab)	Віртуалізований (віддалений МК ATmega2560)
1	2	3
Результати навчання	Заснований на кінестетичному навчанні. Студент фізично взаємодіє з обладнанням, збирає схеми, усвідомлює зв'язок типу «пін – дріт – інтерфейс». Формує базові навички апаратно-програмного інжинірингу	Заснований на абстракції. Оскільки апаратна частина прихована, увага студента повністю концентрується на алгоритмах, моделях скінченних автоматів та розробці програмного коду
Гнучкість та масштабованість	Висока апаратна гнучкість. Відкрита архітектура дозволяє додавати нові електронні компоненти. Можна виконувати легку зміну конфігурації шляхом перемикання фізичних з'єднань	Жорстка інтеграція. Відповідність пінів зафіксовано на рівні модуля керування. Зміна конфігурації можлива виключно на програмному рівні. Додавання зовнішніх модулів неможливе

Продовження таблиці 4.6

1	2	3
Логіка керування та інтерактивність	Інтерактивна / повторювана логіка. Наявність фізичної кнопки дозволяє реалізувати вихід зі стану активного очікування Wait. Користувач повністю контролює частоту та момент запуску кожного циклу	Автономна / кінцева логіка. Відсутність локальних тригерів перетворює стан Wait на кінцевий. Система виконує алгоритм автономно, вимагаючи апаратного Reset для перезапуску
Ресурсомісткість (рівень МК)	Оптимальне використання. ATmega328P (32 КБ Flash, 2 КБ ОЗП) ідеально підходить для масштабу задачі, для якої застосовується 3-х осьовий портал. Ресурси використовуються раціонально без надлишку	Надлишковість. ATmega2560 (256 КБ Flash, 8 КБ ОЗП) має надлишок ресурсів для розробленої програми. Проте це виправдано універсальністю та кількістю портів для забезпечення інших експериментів ВЛ
Енергоефективність (системний рівень)	Низька енергоефективність. Вимагає живлення локального МК та енергозатратного мережевого модуля на стороні кожного окремого користувача	Висока енергоефективність. Реалізує концепцію спільного використання Hardware as a Service. Енергія на модуль керування витрачається на стороні лабораторії, а не на стороні користувача
Розгортання та інтеграція	Локальне апаратно-залежне розгортання. Включає етапи: компіляція коду, локальна прошивка МК через інтерфейс USB, апаратна конфігурація мережевого підключення та встановлення TCP/IP зв'язку з сервером лабораторії	Спрощене віртуалізоване розгортання. Включає етапи: компіляція коду, експорт бінарного файлу у форматі *.hex, завантаження файлу через вебпортал лабораторії та ініціалізація вбудованого МК

Кінець таблиці 4.6

1	2	3
Відмовостійкість та доступність	Низька відмовостійкість. Високий ризик від'єднання дротів на макетній платі. Високий ризик нестабільності підключення кейсу до хмарної інфраструктури або затримок маршрутизації	Висока відмовостійкість. Паяні з'єднання виключають механічні пошкодження. Відсутність постійного мережевого обміну під час виконання експерименту гарантує захист від розривів зв'язку

Проведений аналіз доводить, що кожен з підходів вирішує різні інженерні та освітні завдання.

Підхід на базі апаратного кейса з Arduino UNO є ідеальним інструментом для швидкого створення прототипу та початкових етапів навчання студентів. Він надає розробнику максимальну гнучкість і свободу дій з апаратним забезпеченням та необхідну інтерактивність. Однак, противагою такій гнучкості виступає нижча відмовостійкість та низька енергоефективність через необхідність підтримки роботи локального обладнання.

Підхід на базі вбудованого контролера ATmega2560 демонструє парадигму промислових вбудованих систем та концепцію Hardware as a Service. Тут, замість апаратної гнучкості, віддається перевага жорсткій, але надійній архітектурі. Цей підхід також гарантує високу стабільність виконання алгоритмів. З точки зору сучасних вимог до екологічності та енергозбереження, перенесення обчислювального ядра в загальну хмарну / лабораторну інфраструктуру усуває потребу в дублюванні апаратних засобів у кожного студента, що робить цей підхід глобально більш енергоефективним.

4.5 Висновки до розділу 4

В цьому розділі описано апаратну побудову та логіку функціонування віддаленого експериментального стенда у вигляді 3х-осьового порталу. Розглянуто структуру розробленого ПЗ та ключові особливості його практичної реалізації в середовищі Arduino IDE. Проведено функціональне тестування розробленого ПЗ, що підтвердило коректну роботу на реальному обладнанні ВЛ. Здійснено порівняльний аналіз апаратно-орієнтованого та віртуалізованого підходів до взаємодії з ВЛ та інтегрованого опрацювання даних відалених експериментів, визначено їх переваги, недоліки та доцільність використання з огляду на енерго- та ресурсоефективність, що є важливими з точки зору зелених ІТ.

ВИСНОВКИ

У дипломній роботі виконано дослідження та розробку моделей, способів та програмного забезпечення для інтегрованого опрацювання даних віддаленого експерименту при взаємодії з ВЛ GOLDi 2.0.

На основі аналізу предметної області та інфраструктурних особливостей сучасних ВЛ було ідентифіковано ключові проблеми взаємодії при віддаленому експериментуванні та опрацюванні даних. Дослідження моделей декомпозиції експериментів та сервісно-орієнтованої архітектури ВЛ GOLDi 2.0 дозволило сформувати теоретичне підґрунтя для моделювання процесів взаємодії з ВЛ на основі скінченного автомата Мілі.

Практичну значущість одержаних результатів підтверджено успішною розробкою та тестуванням ПЗ для взаємодії з 3-х осьовим порталом в інфраструктурі ВЛ GOLDi з використанням двох МК сімейства AVR (ATmega328P та ATmega2560) та середовища Arduino IDE для програмної реалізації. Проведені експериментальні дослідження засвідчили працездатність обох реалізованих підходів: на основі локального кейса Hybrid Take-Home-Lab та віддаленого МК, вбудованого в модуль керування.

Виконане на завершальному етапі порівняння двох підходів довело, що використання кейса Hybrid Take-Home-Lab забезпечує гнучкість для швидкого створення прототипу та забезпечує практичні результати навчання, тоді як використання віддаленого МК у складі модуля керування гарантує відмовостійкість, автономність та відповідає сучасним вимогам енергоефективності. Загалом, розроблені моделі взаємодії дозволяють ефективно опрацьовувати дані під час роботи з ВЛ та балансувати між результатами навчання та сучасними вимогами щодо ресурсо- та енергоефективності зелених ІТ.

Результати дослідження було представлено на Щорічній НПК викладачів, науковців, молодих учених, аспірантів та студентів НУ «Запорізька політехніка» «Тиждень науки-2026» (13-17 квітня 2026 року).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Murphy M. P. A. Covid-19 and emergency elearning: Consequences of the securitization of higher education for post-pandemic pedagogy. *Contemporary Security Policy*. 2020. Vol. 41, no. 3. P. 492-505.

2. Online delivery of teaching and laboratory practices: Continuity of university programmes during Covid-19 pandemic / K. A. A. Gamage et. al. *Education Sciences*. 2020. Vol. 10, no. 10. P. 1-9.

3. Adaptable digital labs - motivation and vision of the CrossLab project / I. Aubel et al. *2022 IEEE German Education Conference (GeCon)*: proceedings, 11-12 August 2022, Berlin, Germany / Los Alamitos: IEEE, 2022. P. 1-6.

4. Feisel L. D., Rosa A. J. The role of the laboratory in undergraduate engineering education. *Journal of engineering Education*. 2005. Vol. 94, no. 1. P. 121-130.

5. Berendes J., Gutmann M. Wozu labor? Zur vernachl¨assigten erkenntnistheorie hinter der labordidaktik. *Labore in der Hochschullehre: Labordidaktik, Digitalisierung, Organisation* / Eds. C. Terkowsky et al. 2020. P. 35-49.

6. Satterthwait D. Why are ‘hands-on’ science activities so effective for student learning? *Teaching Science*. 2010. Vol. 56, no. 2. P. 7-10.

7. Велика А., Гарвасюк О., Кропельницька Ю. Роль студентоорієнтованого навчання у підготовці майбутніх фахівців-фармацевтів у вищій школі. *Український педагогічний журнал*. 2023. № 4. С. 74-80.

8. Re-design eines laborpraktikums im lehramtsstudium–didaktische optimierung mittels design-based research / S. Frye et. al. *Labore in der Hochschullehre: Didaktik, Digitalisierung, Organisation*. 2020. P. 95-108.

9. Terkowsky C., Haertel T. Where have all the inventors gone? fostering creativity in engineering education with remote lab learning environments. *IEEE Global Engineering Education Conference (EDUCON)*:

proceedings, 13–15 March 2013, Berlin, Germany / Los Alamitos: IEEE, 2013. P. 345-351.

10. Ellahi R. M., Moin U. A. K., Shah A. Redesigning curriculum in line with Industry 4.0. *Procedia Computer Science*. 2019. Vol. 151. P. 699-708.

11. Henke K., Vietzke T., Hutschenreuter R., Wuttke H.-D. The remote lab cloud "goldi-labs.net". *Remote Engineering and Virtual Instrumentation (REV)*: proceedings of 13th International conference, 24-26 February 2016, Madrid, Spain / Los Alamitos: IEEE, 2016. P. 37–42.

12. Die digitale zukunft des lernens und lehrens mit remote-laboren / T. R. Ortelt et al. *Digitalisierung in Studium und Lehre gemeinsam gestalten*. Wiesbaden: Springer VS, 2021. P. 553–575.

13. Troger P., Rasche A., Feinbube F., Wierschke R. SOA meets robots-a service-based software infrastructure for remote laboratories. *International Journal of Online and Biomedical Engineering*. 2008. Vol. 4, no. 2. P. 57–62.

14. Architectures and design methodologies for scalable and sustainable remote laboratory infrastructures / J. L. Thames et.al. *Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines*. / IGI Global. 2012. Chapter 13. P. 254–275.

15. Grega W. Hardware-in-the-loop simulation and its application in control education. *Designing the Future of Science and Engineering Education*: proceedings of 29th Annual Frontiers in education conference, 10-13 November 1999, San Juan, PR, USA /Los Alamitos: IEEE, 1999. Vol. 2. P. 12B6/7-12B6/12.

16. Dietrich A. Liascript: a domain-specific-language for interactive online courses. *International Conference on e-Learning*: proceedings, 16-19 June 2019, Porto, Portugal, 2019. P. 186–194.

17. Henke K., Vietzke T., Wuttke H.-D., Ostendorff S. Goldi – grid of online lab devices Ilmenau. *International Journal of Online and Biomedical Engineering (iJOE)*. 2016. Vol. 12, no. 04. P. 11–13.

18. Nau J., Henke K., Streitferdt D. New ways for distributed remote web experiments. *Learning with Technologies and Technologies in Learning* / Eds.

Auer, M.E., Pester, A., May, D. Lecture Notes in Networks and Systems. / Springer, Cham. 2022. Vol. 456. P. 257–284.

19. Henke K., Nau J., Streitferdt D. Hybrid take-home labs for the STEM education of the future. *Smart Education and e-Learning - Smart Pedagogy*. *SEEL-22* / Eds. V.L. Uskov, R.J. Howlett, L.C. Jain. Smart Innovation, Systems and Technologies / Springer, Singapore. 2022. Vol. 305. P. 17–26.

20. On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories/ I. Gustavsson et al. *IEEE Transactions on learning technologies*. 2009. Vol. 2, no. 4. P. 263–274.

21. Using VISIR remote lab in the classroom: Case of study of the University of Deusto 2009–2019 / J. Garcia-Zubia et. al. *Cross Reality and Data Science in Engineering. REV 2020* / Eds. M. Auer, D. May. Advances in Intelligent Systems and Computing. / Springer, Cham. 2021. Vol. 1231. P. 82-102.

22. Weblablib: new approach for creating remote laboratories / P. Orduna et al. *Cyber-physical Systems and Digital Twins. REV2019* / Eds. M. Auer, B. K. Ram. Lecture Notes in Networks and Systems / Springer, Cham. 2020. Vol. 80. P. 477–488.

23. Nau J., Henke K. GOLDi Labs as fully integrated learning environment. *2024 IEEE World Engineering Education Conference (EDUNINE)*: proceedings. 10-13 March 2024, Guatemala City, Guatemala / Los Alamitos: IEEE. 2024. P. 1-6.

24. IEEE Standard for networked smart learning objects for online laboratories. IEEE Std 1876-2019. IEEE. 2019. P. 1-57.

25. IEEE Standard for learning object metadata. IEEE Std 1484.12.1-2020. IEEE. 2020. P. 1-50.

26. Henke K., Nau J., Bock R. GOLDi 2.0: Beyond raw digital signals – electrical interface emulation / *Online Engineering and Society 4.0. REV2021* / Eds. M. E. Auer et al. Lecture Notes in Networks and Systems / Springer, Cham. 2021. Vol. 298. P. 23-34.

27. LTI External tools. URL:

https://docs.moodle.org/501/en/LTI_External_tools#Adding_LTI_External_tools_at_course_level (date of access: 15.04.2026).

28. “Hidden” integration of industrial design-tools in e-learning environments / K. Henke et al. *Cross Reality and Data Science in Engineering. REV 2020.* / Eds. M. Auer, D. May. Advances in Intelligent Systems and Computing / Springer, Cham. 2021. Vol. 1231. P. 437-455.

29. Microchip Studio for AVR® and SAM devices. URL: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio (date of access: 15.04.2026).

30. Arduino VENTUNO Q: Where AI takes action. URL: https://www.arduino.cc (date of access: 15.04.2026).

31. Integral Remote laboratory for Programmable Logic / I. Angulo et al. 2019 5th Experiment International Conference (exp.at'19): proceedings, 12-14 June 2019, Funchal, Portugal / Los Alamitos: IEEE, 2019. P. 253-255.

32. GOLDi Labs. Real labs at your home. URL: https://www.goldi-labs.de. (date of access: 15.04.2026).

33. Essentials of microcontroller use learning about peripherals: GPIO. URL: https://www.renesas.com/en/support/engineer-school/mcu-programming-peripherals-01-gpio?srsltid=AfmBOopHJjXGiYJ1Yi1d9CipCFBfvtEwof_19F4Opj4kOV5ZBewRxOpm (date of access: 15.04.2026).

34. Що таке GPIO? URL: https://www.sinsmarts.com/uk/blog/what-is-a-gpio/?srsltid=AfmBOoqdozVzU8UbukJYwPBQJsl6DOppH_ixvSG0o-o5OMvxA_eu5e04 (дата звернення: 15.04.2026).

35. Arduino урок 3 – GPIO URL: https://itmaster.biz.ua/electronics/arduino/arduino-gpio.html (дата звернення: 15.04.2026).

36. Arduino compatible MCU. Our Arduino Compatibles: Nano, Uno, Mega, Pro Mini and Mini. What is Arduino? URL: https://www.az-delivery.de/en/collections/arduino-kompatible-

boards?srsltid=AfmBOoqrIVOpVz3ZMzj63d1KBIZTV_EP4-CfX8xOVjtyAE3lxuVS03zP (date of access: 15.04.2026).

37. The perfect microcontrollers for your project | AZ Delivery. URL: https://www.az-delivery.de/en/collections/mikrocontroller?srsltid=AfmBOoq8h_okB8zSEFCdSVzMq19WdIjE3RStUbYVZ3KpJPz1wD_KNr0g (date of access: 15.04.2026).

38. Colucci A. Arduino boards comparison: A selection guideline choosing the right Arduino board for your project success. URL: https://www.pleasedontcode.com/blog/arduino-boards-comparison-a-selection-guideline (date of access: 15.04.2026).

39. ATmega328P vs ATmega2560: Which microcontroller should you use? URL: https://www.flywing-tech.com/blog/atmega328p-vs-atmega2560-which-microcontroller-should-you-use/ (date of access: 15.04.2026).

40. ATMEGA328P-AUR vs ATMEGA2560-16AUR comparison. URL: https://www.eiyu.com/comparison/atmega328p-aur_atmega2560-16aur#:~:text=Package%20TQFP%2D100%20TQFP%2D32,OscillatorType%20Internal%20Internal (date of access: 15.04.2026).

41. Differences between the two major Arduino boards, UNO and Mega2560. URL: https://www.xiaorgeek.net/blogs/news/differences-between-the-two-major-arduino-boards-uno-and-mega2560#:~:text=The%20main%20upgrade%20of%20UNO,and%20can%20contain%20larger%20programs.&text=We%20can%20see%20that%20the,add%20a%20lot%20of%20expandability (date of access: 15.04.2026).

42. What is the difference between OOP and a structured language? URL: https://www.quora.com/What-is-the-difference-between-OOP-and-a-structured-language (date of access: 15.04.2026).

43. Arduino Object Oriented Programming (OOP). URL: https://roboticsbackend.com/arduino-object-oriented-programming-oop/ (date of access: 15.04.2026).

44. Difference between Structured Programming and Object Oriented Programming. URL: https://www.geeksforgeeks.org/computer-networks/difference-between-structured-programming-and-object-oriented-programming/#google_vignette (date of access: 15.04.2026).

45. Dunbar N. Alternatives to the Arduino IDE. *Arduino Software Internals*. *Maker Innovations Series* / Apress, Berkeley, CA. 2024. P. 163–217.

46. Azzola F. 10 Arduino IDE alternatives to start programming. URL: https://dzone.com/articles/10-arduino-ide-alternatives-to-start-programming (date of access: 15.04.2026).

47. 3-Axis-Portal. URL: https://www.goldi-labs.net/index.php?Site=21 (date of access: 15.04.2026).

48. Цифрова трансформація та «зелені» технології в пріоритеті: ЄС виділив 1,4 млрд євро на підтримку проривних інновацій у 2026 році. URL: https://lnk.ua/EF4rkHf8M (дата звернення: 30.04.2026).

49. Тренд на «зелені» дата-центри у світі: чому і як бізнес інвестує в екорішення. URL: https://hub.kyivstar.ua/articles/trend-na-zeleni-data-czentri-u-sviti-chomu-i-yak-biznes-investuye-v-ekorishennya (дата звернення: 30.04.2026).

50. Guthrie C. How green is Green IT? A multidisciplinary bibliometric study and research agenda. *Procedia Computer Science*. 2024. Vol. 239. P. 701-709.

51. Clifton M. Green IT: An analysis of eco-friendly information technology strategies. *International Journal of Green Technology*. 2019. Vol. 5(1). P. 64-67.

ДОДАТОК А

Текст програми

A.1 Код для Arduino UNO

```
// SENSORS

#define Button A3       // White

#define LimitLeft 2     // Green - D00
#define LimitRight 3    // Blue - D01
#define LimitBack 4     // Violet - D02
#define LimitFront 5    // Gray - D03
#define LimitDown 6     // Orange - D04
#define LimitUp 7       // Yellow - D05
#define Proximity 8     // Brown - D06

// ACTUATORS

#define DriveLeft 9     // Red - D07
#define DriveRight 10   // Brown - D08
#define DriveBack 11    // Orange - D09
#define DriveFront 12   // Yellow - D10
#define DriveDown A0    // Gray - D11
#define DriveUp A1      // Orange - D12
#define Magnet  A2      // Red - D13

// STATES

typedef enum
{
     // Waiting for user button input (try 3-4 seconds after
start)
    Wait,
    // To the back-left corner
    StraightBackLeft,
    StraightLeftBack,
    StraightDiagonal,
    // To the front-right corner
    ReverseFrontRight,
    ReverseRightFront,
    ReverseDiagonal,
    // Lowering and raising
    Down,
    Up,
    // Magnet toggle
    Seize,
    // Returning to default state (top-front-right corner)
    Homing
}
States_t;

// VARIABLES

States_t state;
```

```
int way;  // way = 1 is for straight movement, 2 is for reverse

// RANDOMIZER

void choosePath()
{
    randomSeed(micros());
    int path = random(1, 4);  // different paths from 1 to 3

    if (path == 1)  // diagonal movement
    {
        state = (way == 1) ? StraightDiagonal : ReverseDiagonal;
    }
    else if (path == 2)  // first back then left OR first front
then right
    {
            state = (way == 1) ? StraightBackLeft :
ReverseFrontRight;
    }
    else  // first left then back OR first right then front
    {
            state = (way == 1) ? StraightLeftBack :
ReverseRightFront;
    }
}

// INITIALIZING

void setup()
{
    pinMode(LimitRight, INPUT_PULLUP);
    pinMode(LimitLeft, INPUT_PULLUP);
    pinMode(LimitBack, INPUT_PULLUP);
    pinMode(LimitFront, INPUT_PULLUP);
    pinMode(LimitDown, INPUT_PULLUP);
    pinMode(LimitUp, INPUT_PULLUP);
    pinMode(Proximity, INPUT_PULLUP);
    pinMode(Button, INPUT);

    pinMode(DriveRight, OUTPUT);
    pinMode(DriveLeft, OUTPUT);
    pinMode(DriveBack, OUTPUT);
    pinMode(DriveFront, OUTPUT);
    pinMode(DriveDown, OUTPUT);
    pinMode(DriveUp, OUTPUT);
    pinMode(Magnet, OUTPUT);

    digitalWrite(DriveFront, LOW);
    digitalWrite(DriveBack, LOW);
    digitalWrite(DriveRight, LOW);
    digitalWrite(DriveLeft, LOW);
    digitalWrite(DriveDown, LOW);
    digitalWrite(DriveUp, LOW);
```

```
    digitalWrite(Magnet, LOW);

    state = Homing;
    way = 1;

    delay(2000);
}

// UPDATING

void loop()
{
    switch (state)
    {
        case Wait:
        {
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);
            digitalWrite(Magnet, LOW);

                                if    (!digitalRead(LimitUp)    ||
!digitalRead(LimitFront) || !digitalRead(LimitRight))
            {
                state = Homing;
                break;
            }

            if (!digitalRead(Button))
            {
                choosePath();
            }
            break;
        }

        case StraightDiagonal:
        {
            digitalWrite(DriveBack, !digitalRead(LimitBack));
            digitalWrite(DriveLeft, !digitalRead(LimitLeft));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                               if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
```

```
            break;
        }

        case StraightBackLeft:
        {
            digitalWrite(DriveBack, !digitalRead(LimitBack));
              digitalWrite(DriveLeft, !digitalRead(LimitLeft) &&
digitalRead(LimitBack));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
            break;
        }

        case StraightLeftBack:
        {
            digitalWrite(DriveLeft, !digitalRead(LimitLeft));
              digitalWrite(DriveBack, !digitalRead(LimitBack) &&
digitalRead(LimitLeft));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
            break;
        }


        case ReverseDiagonal:
        {
            digitalWrite(DriveFront, !digitalRead(LimitFront));
            digitalWrite(DriveRight, !digitalRead(LimitRight));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                               if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
```

```
            {
                way = 1;
                state = Down;
            }
            break;
        }

        case ReverseFrontRight:
        {
            digitalWrite(DriveFront, !digitalRead(LimitFront));
             digitalWrite(DriveRight, !digitalRead(LimitRight) &&
digitalRead(LimitFront));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
            {
                way = 1;
                state = Down;
            }
            break;
        }

        case ReverseRightFront:
        {
            digitalWrite(DriveRight, !digitalRead(LimitRight));
             digitalWrite(DriveFront, !digitalRead(LimitFront) &&
digitalRead(LimitRight));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
            {
                way = 1;
                state = Down;
            }
            break;
        }


        case Down:
        {
              digitalWrite(DriveDown, !digitalRead(LimitDown) &&
!digitalRead(Proximity));
            digitalWrite(DriveUp, LOW);
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
```

```
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

                                if    (digitalRead(LimitDown)    ||
digitalRead(Proximity))
            {
                state = Seize;
            }
            break;
        }

        case Up:
        {
            digitalWrite(DriveUp, !digitalRead(LimitUp));
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

            if (digitalRead(LimitUp))
            {
                if (way == 2)
                {
                    choosePath();
                }
                else if (digitalRead(Button))
                {
                    state = Wait;
                }
            }
            break;
        }

        case Seize:
        {
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

            if (way == 2)
            {
                digitalWrite(Magnet, HIGH);
            }
            else
            {
                digitalWrite(Magnet, LOW);
            }

            delay(1000);
```

```
            state = Up;
            break;
        }

        case Homing:
        {
             if (digitalRead(LimitUp) && digitalRead(LimitFront)
&& digitalRead(LimitRight))
            {
                state = Wait;
                break;
            }

            digitalWrite(DriveUp, !digitalRead(LimitUp));
             digitalWrite(DriveFront, !digitalRead(LimitFront) &&
digitalRead(LimitUp));
             digitalWrite(DriveRight, !digitalRead(LimitRight) &&
digitalRead(LimitUp));
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(Magnet, LOW);
            break;
        }
    }
}
```

A.2 Код для ATmega2560

```
// SENSORS

#define LimitLeft 3     // PE5
#define LimitRight 2    // PE4
#define LimitBack 4     // PG5
#define LimitFront 5    // PE3
#define LimitDown 13    // PB7
#define LimitUp 14      // PJ1
#define Proximity 15    // PJ0

// ACTUATORS

#define DriveLeft 8     // PH5
#define DriveRight 7    // PH4
#define DriveBack 9     // PH6
#define DriveFront 10   // PB4
#define DriveDown 11    // PB5
#define DriveUp 12      // PB6
#define Magnet 30       // PC7

// STATES
```

```
typedef enum
{
    // Waiting
    Wait,
    // To the back-left corner
    StraightBackLeft,
    StraightLeftBack,
    StraightDiagonal,
    // To the front-right corner
    ReverseFrontRight,
    ReverseRightFront,
    ReverseDiagonal,
    // Lowering and raising
    Down,
    Up,
    // Magnet toggle
    Seize,
    // Returning to default state (top-front-right corner)
    Homing
}
States_t;

// VARIABLES

States_t state;
int way;  // way = 1 is for straight movement, 2 is for reverse

// RANDOMIZER

void choosePath()
{
    randomSeed(micros());
    int path = random(1, 4);  // different paths from 1 to 3

    if (path == 1)  // diagonal movement
    {
        state = (way == 1) ? StraightDiagonal : ReverseDiagonal;
    }
    else if (path == 2)  // first back then left OR first front
then right
    {
            state = (way == 1) ? StraightBackLeft :
ReverseFrontRight;
    }
    else  // first left then back OR first right then front
    {
            state = (way == 1) ? StraightLeftBack :
ReverseRightFront;
    }
}

// INITIALIZING
```

```
void setup()
{
    pinMode(LimitRight, INPUT_PULLUP);
    pinMode(LimitLeft, INPUT_PULLUP);
    pinMode(LimitBack, INPUT_PULLUP);
    pinMode(LimitFront, INPUT_PULLUP);
    pinMode(LimitDown, INPUT_PULLUP);
    pinMode(LimitUp, INPUT_PULLUP);
    pinMode(Proximity, INPUT_PULLUP);

    pinMode(DriveRight, OUTPUT);
    pinMode(DriveLeft, OUTPUT);
    pinMode(DriveBack, OUTPUT);
    pinMode(DriveFront, OUTPUT);
    pinMode(DriveDown, OUTPUT);
    pinMode(DriveUp, OUTPUT);
    pinMode(Magnet, OUTPUT);

    digitalWrite(DriveFront, LOW);
    digitalWrite(DriveBack, LOW);
    digitalWrite(DriveRight, LOW);
    digitalWrite(DriveLeft, LOW);
    digitalWrite(DriveDown, LOW);
    digitalWrite(DriveUp, LOW);
    digitalWrite(Magnet, LOW);

    state = Homing;
    way = 1;

    delay(7000);
}

// UPDATING

void loop()
{
    switch (state)
    {
        case Wait:
        {
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);
            digitalWrite(Magnet, LOW);

            break;
        }

        case StraightDiagonal:
```

```
        {
            digitalWrite(DriveBack, !digitalRead(LimitBack));
            digitalWrite(DriveLeft, !digitalRead(LimitLeft));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
            break;
        }

        case StraightBackLeft:
        {
            digitalWrite(DriveBack, !digitalRead(LimitBack));
              digitalWrite(DriveLeft, !digitalRead(LimitLeft)  &&
digitalRead(LimitBack));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
            break;
        }

        case StraightLeftBack:
        {
            digitalWrite(DriveLeft, !digitalRead(LimitLeft));
              digitalWrite(DriveBack, !digitalRead(LimitBack)  &&
digitalRead(LimitLeft));
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitBack)    &&
digitalRead(LimitLeft))
            {
                way = 2;
                state = Down;
            }
            break;
```

```
        }


        case ReverseDiagonal:
        {
            digitalWrite(DriveFront, !digitalRead(LimitFront));
            digitalWrite(DriveRight, !digitalRead(LimitRight));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
            {
                way = 1;
                state = Down;
            }
            break;
        }

        case ReverseFrontRight:
        {
            digitalWrite(DriveFront, !digitalRead(LimitFront));
             digitalWrite(DriveRight, !digitalRead(LimitRight) &&
digitalRead(LimitFront));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
            {
                way = 1;
                state = Down;
            }
            break;
        }

        case ReverseRightFront:
        {
            digitalWrite(DriveRight, !digitalRead(LimitRight));
             digitalWrite(DriveFront, !digitalRead(LimitFront) &&
digitalRead(LimitRight));
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);

                                if    (digitalRead(LimitFront)    &&
digitalRead(LimitRight))
            {
```

```
                way = 1;
                state = Down;
            }
            break;
        }


        case Down:
        {
              digitalWrite(DriveDown, !digitalRead(LimitDown) &&
!digitalRead(Proximity));
            digitalWrite(DriveUp, LOW);
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

                                if    (digitalRead(LimitDown)    ||
digitalRead(Proximity))
            {
                state = Seize;
            }
            break;
        }

        case Up:
        {
            digitalWrite(DriveUp, !digitalRead(LimitUp));
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveFront, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

            if (digitalRead(LimitUp))
            {
                if (way == 2)
                {
                    choosePath();
                }
                else
                {
                    state = Wait;
                }
            }
            break;
        }

        case Seize:
        {
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveUp, LOW);
            digitalWrite(DriveFront, LOW);
```

```
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveRight, LOW);
            digitalWrite(DriveLeft, LOW);

            if (way == 2)
            {
                digitalWrite(Magnet, HIGH);
            }
            else
            {
                digitalWrite(Magnet, LOW);
            }

            delay(1000);

            state = Up;
            break;
        }

        case Homing:
        {
             if (digitalRead(LimitUp) && digitalRead(LimitFront)
&& digitalRead(LimitRight))
            {
                digitalWrite(DriveFront, LOW);
                digitalWrite(DriveBack, LOW);
                digitalWrite(DriveRight, LOW);
                digitalWrite(DriveLeft, LOW);
                digitalWrite(DriveDown, LOW);
                digitalWrite(DriveUp, LOW);
                digitalWrite(Magnet, LOW);

                choosePath();
                delay(3500);
                break;
            }

            digitalWrite(DriveUp, !digitalRead(LimitUp));
             digitalWrite(DriveFront, !digitalRead(LimitFront) &&
digitalRead(LimitUp));
             digitalWrite(DriveRight, !digitalRead(LimitRight) &&
digitalRead(LimitUp));
            digitalWrite(DriveDown, LOW);
            digitalWrite(DriveBack, LOW);
            digitalWrite(DriveLeft, LOW);
            digitalWrite(Magnet, LOW);
            break;
        }
    }
}
```

ДОДАТОК Б

Слайди презентації

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

Розроблення та дослідження моделей і способів опрацювання даних при взаємодії з віддаленою лабораторією

Виконав
студент гр. КНТ-212 Ілля ПАРХОМЕНКО
Керівник проєкту
к.т.н., професор Галина ТАБУНЩИК

Рисунок Б.1 – Слайд 1

Актуальність роботи

Місце Hybrid Take-Home-Lab в класифікації онлайн лабораторій

Гібридна лабораторія GOLDi 2.0

Джерело: https://www.goldi-labs.de

2

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

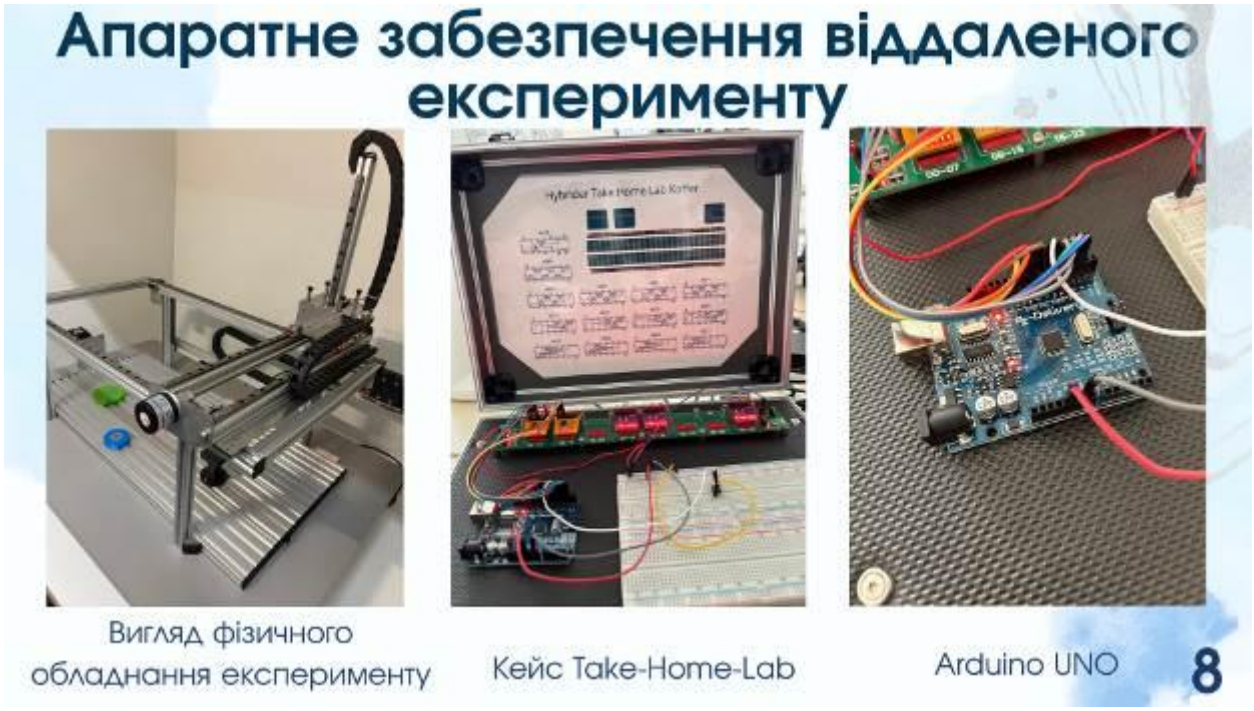


Рисунок Б.8 – Слайд 8

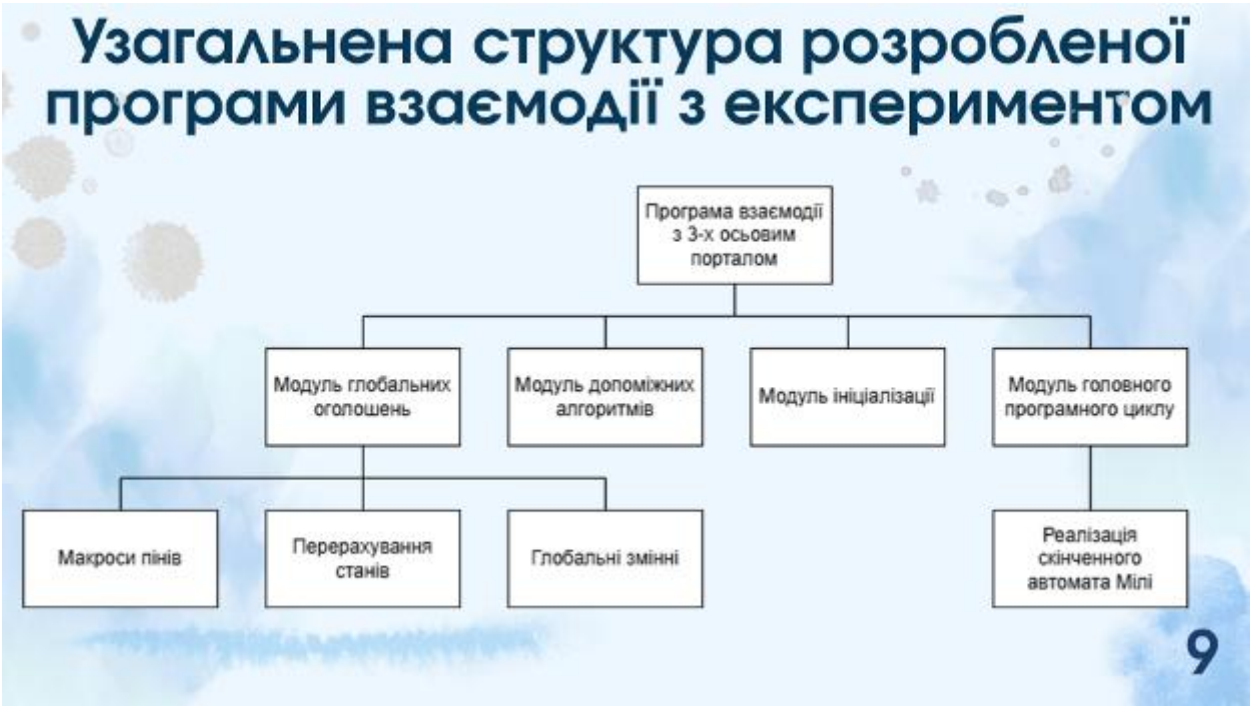


Рисунок Б.9 – Слайд 9

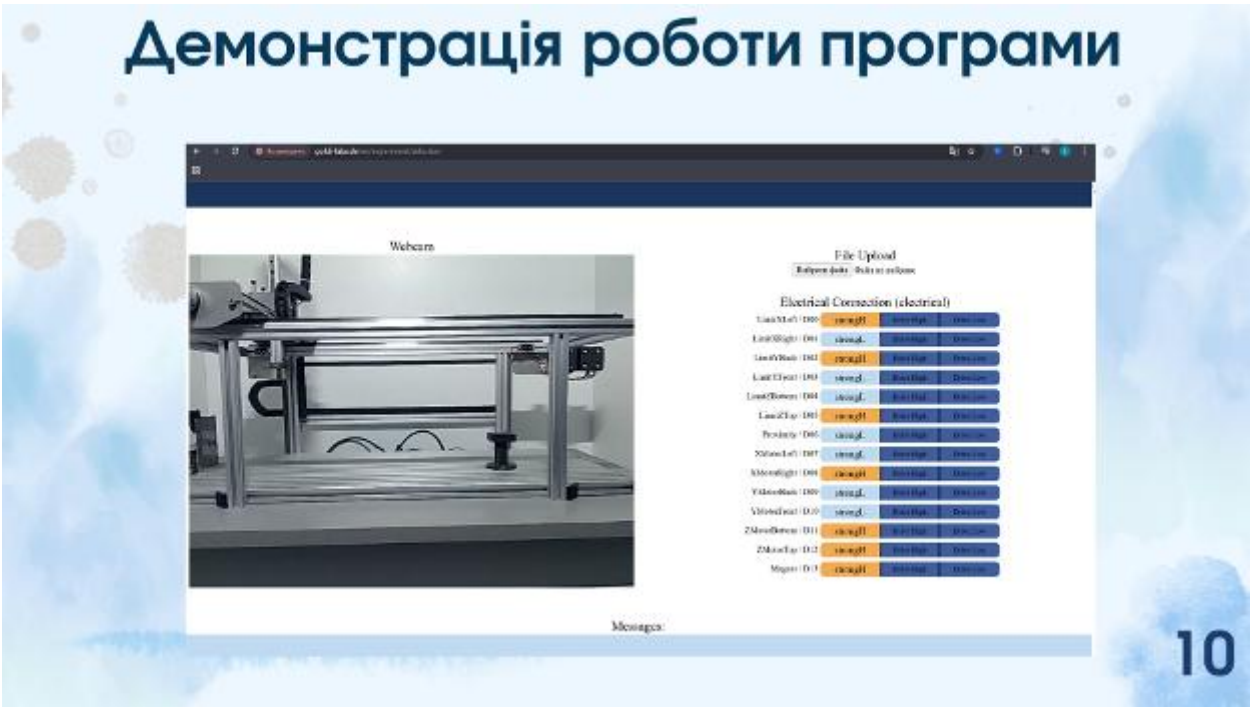


Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

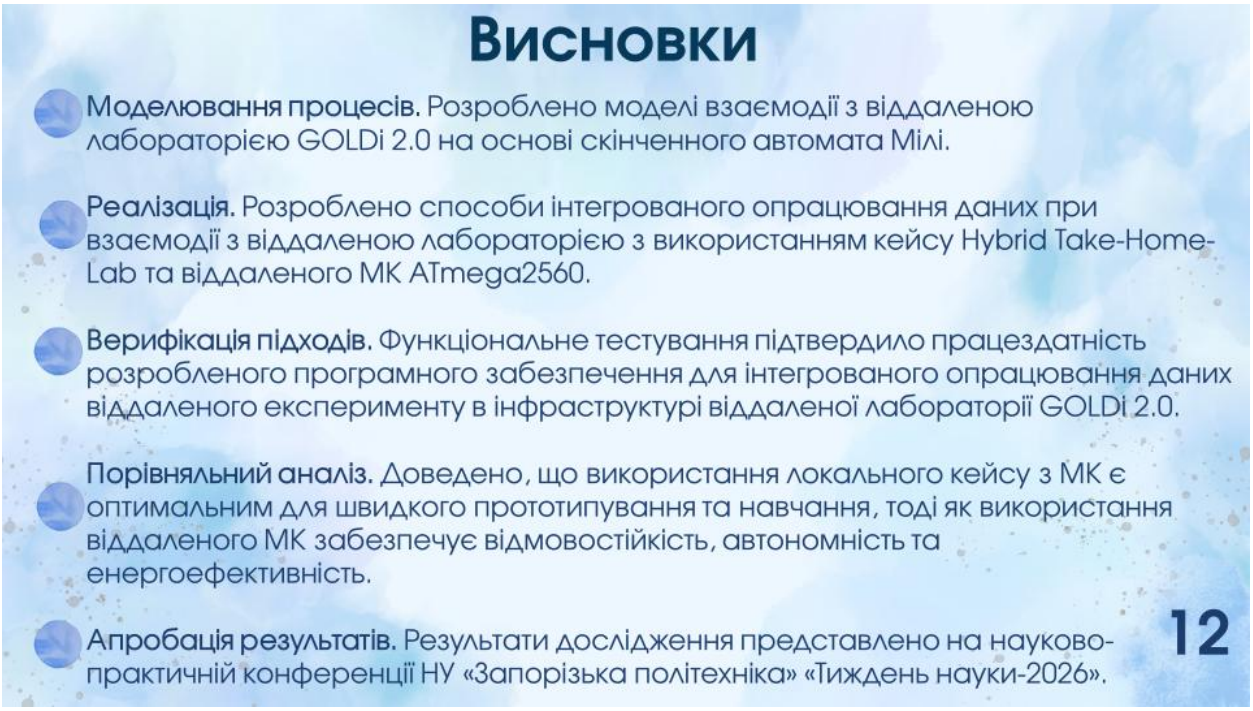


Рисунок Б.12 – Слайд 12