

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 ПРОГРАМА ДЛЯ МОДЕЛЮВАННЯ НЕБІНАРНИХ

(назва теми)

АВТОМАТІВ

SOFTWARE TO MODEL NON-BINARY AUTOMATA

Виконав: студент(ка) 4 курсу, групи КНТ-210сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ШИКІНА Є.А.

(ПРІЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТИМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

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

Факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки і технології

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки

(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2023 року

**З А В Д А Н Н Я**

**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

ШИКІНОЇ Єлизавети Андріївни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програма для моделювання небінарних автоматів.

Software to Model Non-binary Automata.

керівник проєкту (роботи) д.т.н., професор, ПОЛЯКОВ Михайло Олексійович,  
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 16 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програмного забезпечення. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | Прізвище, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ПОЛЯКОВ М.О., професор                    |                |                           |
| Нормоконтроль       | ДЕЙНЕГА Л.Ю., ст. викладач                |                |                           |

7. Дата видачі завдання “ 04 ” квітня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

Керівник проєкту (роботи)

|            |                         |
|------------|-------------------------|
|            | <u>Єлизавета ШИКІНА</u> |
| ( підпис ) | (Ім'я ПРІЗВИЩЕ)         |
|            | <u>Михайло ПОЛЯКОВ</u>  |
| ( підпис ) | (Ім'я ПРІЗВИЩЕ)         |

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 75 с., 20 рис., 3 дод., 14 джерел.

КОМП'ЮТЕРНІ НАУКИ, МОДЕЛЮВАННЯ, КІНЦЕВИЙ НЕБІНАРНИЙ АВТОМАТ, КЕРУВАННЯ ПОДІЯМИ, ІНТЕРАКТИВНА МОДЕЛЬ КІНЦЕВОГО НЕБІНАРНОГО АВТОМАТУ, PYTHON, БІБЛІОТЕКА PYGAME, HTML, CSS.

Об'єкт дослідження – є самі небінарні автомати. В рамках проекту проводиться дослідження їх властивостей, особливостей та можливих застосувань. Розробка програми базується на цих дослідженнях, щоб забезпечити коректну моделювання та виконання операцій з небінарними автоматами.

Предмет дослідження – алгоритми та методи для реалізації небінарних кінцевих автоматів.

Мета роботи – розробка програмного забезпечення, яке здатне моделювати та виконувати операції з небінарними автоматами.

Матеріали, методи та технічні засоби: середовище розробки PyCharm. Основні інструменти для розробки: мова програмування Python, набір функцій (фреймворк) Matplotlib, PyGame.

Результати. Розроблено веб застосунок для моделювання роботи кінцевих автоматів.

Висновки. Результати роботи повністю відповідають поставленій меті та визначеному технічному завданню до проекту. Проведено аналіз та порівняння з існуючими подібними аналогами. Розроблено програмний продукт, який призначений для демонстрації моделі кінцевих автоматів.

Галузь використання – проект може бути використаний у навчальних цілях, щоб продемонструвати різні галузі використання небінарних кінцевих автоматів.

## ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 75 pages, 20 figure, 3 appendixes, 14 sources.

COMPUTER SCIENCE, SIMULATION, NON-BINARY FINITE AUTOMATA, EVENT MANAGEMENT, INTERACTIVE NON-BINARY FINITE AUTOMATA MODEL, PYTHON, PYGAME LIBRARY, HTML, CSS.

The object of research is non-binary automata themselves. As part of the project, a study of their power, features and possible programs is conducted. Development of a program based on these studies to ensure correct modelling and execution of operations with non-binary automata.

The subject of research is algorithms and methods for the implementation of non-binary finite state machines.

The goal of the work is to develop software capable of simulating and performing operations with non-binary automata.

Materials, Methods, and Techniques: PyCharm development environment. Main development tools: Python programming language, set of functions (framework) Matplotlib, PyGame.

Results. A web application for simulating the operation of finite state machines has been developed.

Conclusions. The results of the work fully correspond to the set goal and the defined technical task for the project. An analysis and comparison with existing similar analogues was carried out. A software product designed to demonstrate the finite state machine model has been developed.

Field of use – the project can be used for educational purposes to demonstrate various fields of use of non-binary state machines.

## ЗМІСТ

|                                                                          | С. |
|--------------------------------------------------------------------------|----|
| Перелік скорочень та умовних позначок.....                               | 8  |
| Вступ.....                                                               | 9  |
| 1 Аналіз предметної області.....                                         | 10 |
| 1.1 Опис предметної.....                                                 | 10 |
| 1.2 Аналіз аналогів .....                                                | 14 |
| 1.2.1 «Goldi-labs.net» .....                                             | 14 |
| 1.2.2 «MATLAB».....                                                      | 15 |
| 1.2.3 «SkyCAD Electrical» .....                                          | 17 |
| 1.3 Висновки до розділу 1 .....                                          | 18 |
| 2 Розробка структури програмного забезпечення.....                       | 19 |
| 2.1 Основні засоби розробки програми.....                                | 19 |
| 2.1.1 Мова програмування Python.....                                     | 19 |
| 2.1.2 Мова гіпертекстової розмітки HTML.....                             | 20 |
| 2.1.3 Каскадна таблиця стилів-CSS .....                                  | 21 |
| 2.1.4 PyGame Library.....                                                | 23 |
| 2.2 Вибір середовища розробки .....                                      | 25 |
| 2.3 Висновки до розділу 2.....                                           | 26 |
| 3 Розробка програмного забезпечення .....                                | 27 |
| 3.1 Опис алгоритму .....                                                 | 27 |
| 3.2 Структурна схема проєкту.....                                        | 30 |
| 3.3 Опис реалізованих функцій .....                                      | 31 |
| 3.4 Висновки до розділу 3.....                                           | 34 |
| 4 Експлуатація, тестування та експериментальне дослідження програми..... | 35 |
| 4.1 Призначення програми.....                                            | 35 |
| 4.2 Умови виконання програми.....                                        | 35 |
| 4.3 Тестування в критичних середовищах .....                             | 36 |
| 4.4 Тестування програми .....                                            | 37 |
| 4.5 Висновки до розділу 4.....                                           | 38 |

|                                   |    |
|-----------------------------------|----|
| Висновки.....                     | 39 |
| Перелік джерел посилань .....     | 41 |
| Ддаток А Технічне завдання .....  | 43 |
| Додаток Б Текст програми .....    | 47 |
| Додаток В Слайди презентації..... | 70 |

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

- КА – Кінцевий Автомат;  
FSM – Finite State Machine;  
IDE – Integrated Developer Environment;  
ПЗ – Програмне забезпечення;

## ВСТУП

В сучасному інформаційному суспільстві комп'ютерні системи та програмне забезпечення відіграють важливу роль у багатьох сферах людського життя. Однак, існує клас задач, де традиційні бінарні кінцеві автомати не виявляються достатньо потужними для моделювання поведінки системи. У таких випадках необхідність у використанні небінарних кінцевих автоматів стає очевидною [1].

Цей дипломний проєкт присвячений розробці програми для моделювання небінарних кінцевих автоматів. Програма дозволяє симулювати автомати зі станами, переходами та діями, в яких вхідні символи можуть приймати більше двох значень. Це відкриває широкі можливості для моделювання різноманітних сценаріїв та поведінки систем.

Зважаючи на зростаючу потребу у моделюванні небінарних кінцевих автоматів у різних галузях, цей дипломний проєкт має актуальність і практичну значущість. Розробка програми для моделювання небінарних кінцевих автоматів відкриває шлях до вдосконалення процесу проектування та аналізу складних систем, де використання традиційних бінарних автоматів стає недостатнім.

Метою дипломної кваліфікаційної роботи проєкту є оцінка та аналіз існуючих методів та програмна реалізація застосунку для впровадження моделі небінарних кінцевих автоматів.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі проводиться аналіз предметної області, пов'язаної з небінарними автоматами. Визначаються основні поняття, принципи та методи, які стосуються небінарних автоматів. Досліджуються існуючі програмні засоби, що використовуються для моделювання.

## 1.1 Опис предметної

Кінцеві автомати (КА) або англійською Finite State Machines (FSM) є математичною моделлю, що використовується для опису імовірних послідовностей подій [2].

Основна суть використання FSM полягає в тому, що програма може перебувати у конкретному стані, і залежно від вхідних подій або станів може змінювати свій поточний стан. Кожен стан відповідає певному набору дій, які програма може виконувати у цьому стані, а також переходам у нові стани відповідно до вхідних подій.

Автомати перетворювачі визначаються як кортеж за такою формулою:

$$A = \langle S, X, Y, s_0, \delta, \lambda \rangle, \quad (1.1)$$

де  $S$  — скінченна непорожня множина (станів);

$X$  — скінченна непорожня множина входів (вхідний алфавіт);

$Y$  — скінченна непорожня множина виходів (вихідний алфавіт);

$s_0$  — початковий стан;

$\delta : S \times X \rightarrow S$  — функція переходів;

$\lambda : S \times X \rightarrow Y$ ,  $\lambda : S \rightarrow Y$  — функції виходів автоматів Мілі та Мура відповідно [3].

Кінцеві автомати можуть бути бінарними або небінарними, в залежності від кількості можливих станів.

Бінарні кінцеві автомати мають два можливих стани: «0» і «1». Як варіант ще можуть називатися як «увімкнено» і «вимкнено» або «активно» і «не активно». Їх можна представити за допомогою наступних математичних формул:

- множина станів ( $S$ ):  $S = \{0, 1\}$ ;
- множина вхідних символів ( $I$ ):  $I = \{0, 1\}$ ;
- множина вихідних символів ( $O$ ):  $O$  (може бути пустою, залежно від застосування);
- початковий стан ( $S_0$ ):  $S_0$  (зазвичай "0" або "1");
- функція переходу ( $\delta$ ):  $\delta: S \times I \rightarrow S$  - приймає поточний стан і вхідний символ, повертає наступний стан;
- вихідна функція ( $\gamma$ ):  $\gamma: S \times I \rightarrow O$  (залежно від застосування виводить вихідні дані).

Натомість, небінарні кінцеві автомати можуть мати більше ніж два стани. Вони представляються аналогічними формулами, але множини станів, вхідних символів та вихідних символів можуть містити більше значень. Наприклад:

- множина станів ( $S$ ):  $S = \{s_1, s_2, \dots, s_n\}$ ;
- множина вхідних символів ( $I$ ):  $I = \{i_1, i_2, \dots, i_m\}$ ;
- множина вихідних символів ( $O$ ):  $O$  (залежно від застосування);
- початковий стан ( $S_0$ ):  $S_0$  (один зі станів з множини  $S$ );
- функція переходу ( $\delta$ ):  $\delta: S \times I \rightarrow S$ ;
- вихідна функція ( $\gamma$ ):  $\gamma: S \times I \rightarrow O$  (залежно від застосування).

Функція переходу ( $\delta$ ) визначає, який наступний стан буде у кінцевому автоматі, залежно від поточного стану і вхідного символу. Вихідна функція ( $\gamma$ ) визначає, якій вихідний символ буде згенерований кінцевим автоматом, залежно від поточного стану і вхідного символу.

Модель кінцевого автомата може бути подана у вигляді графа, де стани є вузлами, а переходи між станами - ребрами. Кожен ребро відповідає вхідному символу, і може мати асоційоване значення вихідного символу.

Приклад графа кінцевого автомата представлений на рисунку 1.1 [4].

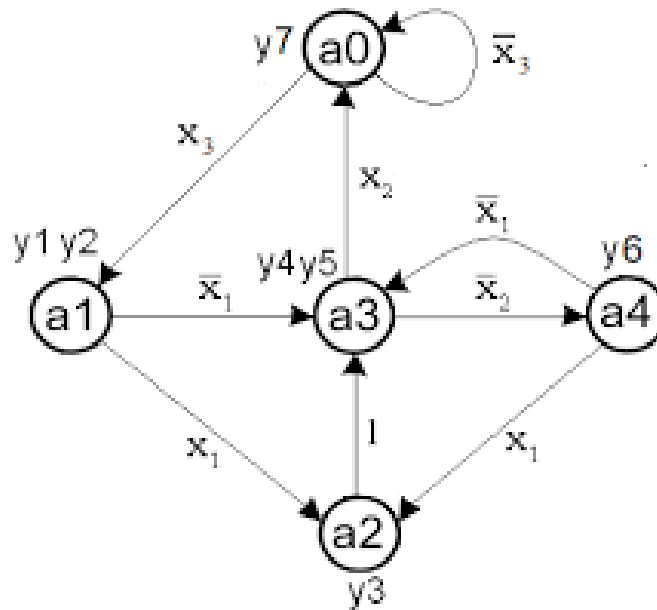


Рисунок 1.1 – Приклад графа кінцевого автомату

Разом з графом іде таблиця переходів станів, представлена нижче на рисунку 1.2.

| № | $a_m$ | $y_m$    | $x_{ms}$         | $a_s$ |
|---|-------|----------|------------------|-------|
| 0 | $a_0$ | $y_7$    | $\overline{x_3}$ | $a_0$ |
|   |       |          | $x_3$            | $a_1$ |
| 1 | $a_1$ | $y_1y_2$ | $\overline{x_1}$ | $a_2$ |
|   |       |          | $x_1$            | $a_3$ |
| 2 | $a_2$ | $y_3$    | 1                | $a_3$ |
| 3 | $a_3$ | $y_4y_5$ | $\overline{x_2}$ | $a_0$ |
|   |       |          | $x_2$            | $a_4$ |
| 4 | $a_4$ | $y_6$    | $\overline{x_1}$ | $a_2$ |
|   |       |          | $x_1$            | $a_3$ |

Рисунок 1.2 – Таблиця переходів станів

FSM застосовується в багатьох різних сферах, оскільки він може

описувати послідовні стани та переходи між ними. Ось деякі з них:

- комп'ютерна програмування: FSM широко використовується для моделювання та управління поведінкою програмного забезпечення. Він застосовується у розробці компіляторів, інтерпретаторів, обробки мови, автоматизованих систем управління, графічних інтерфейсів користувача та багатьох інших областях програмування;

- протоколи мережі: FSM використовується для опису та аналізу протоколів мережі, таких як TCP/IP, HTTP, FTP, SMTP та інших. Він може моделювати різні стани, в яких може перебувати мережевий протокол, та переходи між цими станами в залежності від подій, що відбуваються в мережі;

- апаратне проектування: FSM застосовується для моделювання та управління поведінкою цифрових електронних пристроїв, таких як мікропроцесори, контролери, цифрові системи управління тощо. Він може описувати різні стани та переходи між ними в залежності від вхідних сигналів;

- програмування штучного інтелекту: FSM використовується для моделювання поведінки агентів у системах штучного інтелекту. Він може описувати стани та переходи, пов'язані з діями та реакціями агента на зміни у своєму оточенні;

- контроль та автоматизація: FSM використовується в системах контролю та автоматизації для моделювання та керування послідовністю операцій. Він може описувати стани різних пристроїв та переходи між ними в залежності від умов та вхідних сигналів, що дозволяє ефективно керувати послідовністю робочих процесів;

- проектування протоколів комунікації: FSM використовується для проектування та аналізу протоколів комунікації між різними системами. Він дозволяє описати стани та поведінку системи взаємодії, включаючи передачу даних, обробку помилок та керування послідовністю комунікації;

- розпізнавання та обробка сигналів: FSM використовується в області

обробки сигналів для моделювання та аналізу послідовностей сигналів. Він може описувати стани різних сигналів та переходи між ними, що дозволяє виявляти та аналізувати певні шаблони або поведінку в сигналах;

- графічне моделювання: FSM використовується в графічному моделюванні для опису та управління поведінкою об'єктів та сценаріїв. Він може описувати різні стани об'єктів, їхні взаємодії та переходи між ними, дозволяючи реалізувати взаємодію та поведінку об'єктів у графічному середовищі [5].

## 1.2 Аналіз аналогів

FSM (шаблон кінцевого автомата) також знаходить широке застосування в сучасних технологіях. Наприклад, в онлайн-сервісах для управління завданнями. КА може бути використаний для автоматизації процесу керування завданнями, користувача або станами в наданому процесі.

### 1.2.1 «Goldi-labs.net»

Goldi-labs.net [6] – вебсайт, який надає можливість проектувати і тестувати алгоритми керування для різних систем, таких як ліфт та склад.

На рис. 1.3 наведено приклад демонстрації роботи ліфту в «Goldi-labs.net».

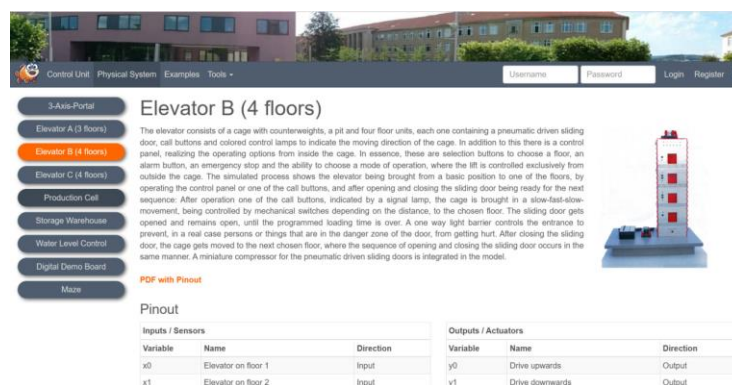


Рисунок 1.3 – Сторінка моделі роботи ліфту у Goldi-labs

Основні переваги цього сервісу включають наступне:

- розширений функціонал: GoldiLabs дозволяє реалізовувати алгоритми керування в різних форматах, включаючи булеві рівняння на основі скінченних автоматів, апаратне або програмне забезпечення;
- гнучкість: Ви можете використовувати мікроконтролери, апаратні платформи FPGA або CPLD, або програмне забезпечення, такі як Assembler або C, для реалізації своїх алгоритмів керування;
- віддалений доступ: GoldiLabs дозволяє вам здійснювати керування навіть у дистанційній лабораторії, що забезпечує зручний віддалений доступ та тестування алгоритмів;
- тестування реальних систем: Ви можете використовувати GoldiLabs для тестування алгоритмів керування не тільки на віртуальних системах, але й на реальних фізичних системах, таких як ліфт або склад.

### 1.2.2 «MATLAB»

Ще одним аналогом є «MATLAB». MATLAB [7]-[8] - система комп'ютерної алгебри з класу автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями та візуальним супроводом.

На рис. 1.4 наведено користувацький інтерфейс програми «MATLAB».

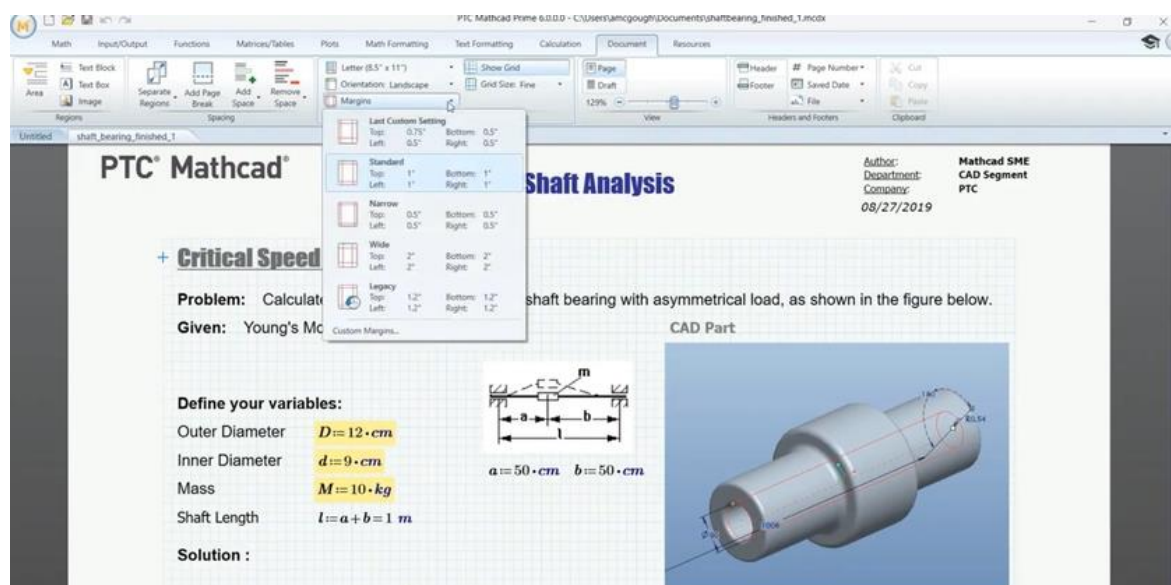


Рисунок 1.4 – Можливий вигляд програми «MATLAB»

Переваги «MATLAB»:

- висока точність при виконанні великого обсягу обчислень;
- зручний у застосуванні та використанні;
- адаптивність: сервіс можливо використовувати як на персональному комп'ютері, так і на смартфоні;
- гнучке налаштування проєктування;
- розширюваність: можливість інтегрувати в інші програми.

Недолік:

- платність програми;
- труднощі в реалізації складних алгоритмів.

### 1.2.3 «SkyCAD Electrical»

SkyCAD Electrical [9] – програма для розробки електронних схем. Він забезпечує високоякісні електричні конструктивні функції, не потребує спеціальної підготовки для освоєння та має розумну ціну. Під стандартною ліцензією даний продукт надає змогу розробляти основні схеми, використовуючи стандартні інструменти, необхідні більшості фабрик для автоматизації.

На рис. 1.5 наведено користувацький інтерфейс програми «SkyCAD Electrical».

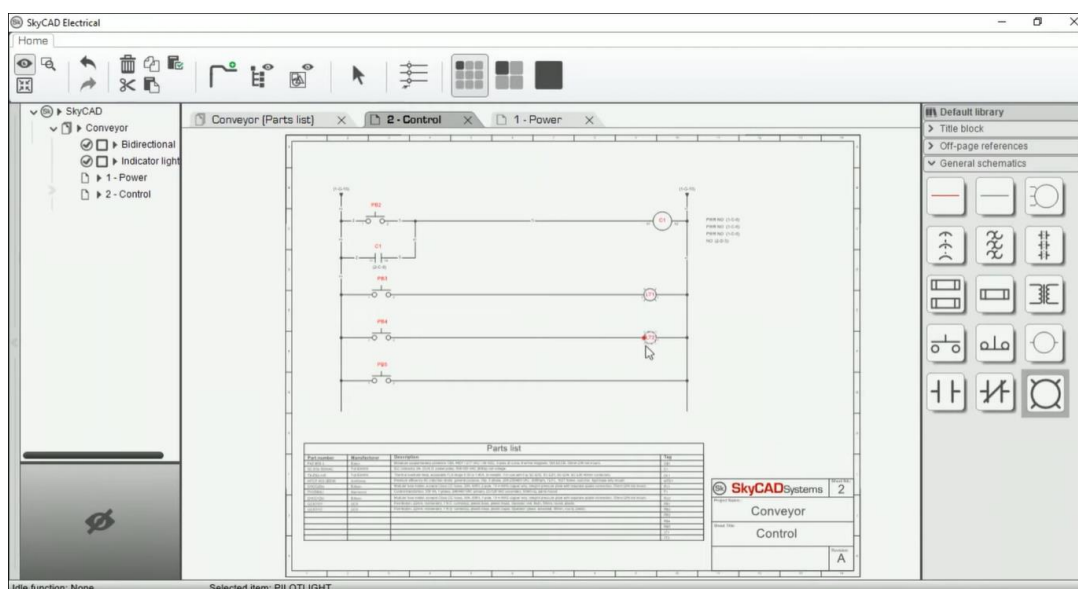


Рисунок 1.5 – Вкладка для тестування розробленої автоматизації у «SkyCAD Electrical»

Переваги:

- можливість створювати власні схеми автоматизації;
- надає звіти у вигляді Excel таблиць;
- наявність матеріалу як від постачальника продукту так і на просторах Інтернету, наявність спілки користувачів.

Недоліки:

- повний доступ до всіх можливостей програми є платним.

### **1.3 Висновки до розділу 1**

Отже, кінцеві автомати знаходять широке застосування в різних галузях, таких як розробка ігор, вбудовані системи, комп'ютерні науки, мережеві протоколи, телекомунікації, автоматизація, (наприклад, тестування) та інші. Вони є потужним інструментом для моделювання послідовностей подій і можуть бути використані для розв'язання різноманітних завдань, наприклад, розпізнавання мови, синтаксичного аналізу, керування вхідно-вихідними пристроями, управління меню або станами персонажів в іграх та іншого.

## **2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

У даному розділі буде розглянуто особливості програмних засобів для реалізації дипломної кваліфікаційної роботи.

### **2.1 Основні засоби розробки програми**

#### **2.1.1 Мова програмування Python**

Python [10] є високорівневою, інтерпретованою мовою програмування, яка володіє простим і зрозумілим синтаксисом. Вона була розроблена у 1990-х роках і з тих пір стала однією з найпопулярніших мов для розробки програм. Python має широке застосування у різних галузях, включаючи веб-розробку, наукові обчислення, штучний інтелект, аналіз даних та багато іншого.

Використання Python для описання моделі небінарних кінцевих автоматів є доцільним з кількох причин. По-перше, Python має простий і зрозумілий синтаксис, що дозволяє легко виразити логіку та переходи між станами автомата. Крім того, в Python існує багато бібліотек і фреймворків, які спрощують роботу з автоматами, забезпечуючи готові рішення для реалізації їх функціональності. Такі бібліотеки стануть в нагоді під час розширення проєкту, або при описі автоматизації більш складних систем.

Python також має кілька переваг, які роблять його привабливим для моделювання небінарних кінцевих автоматів. По-перше, він є мультиплатформеним, що означає, що програми, написані на Python, можуть працювати на різних операційних системах. Крім того, Python має велику спільноту розробників, яка активно підтримує мову та надає багато корисних ресурсів та документацію.

Незважаючи на свої переваги, Python також має деякі недоліки. По-перше, порівняно з іншими мовами програмування, Python може бути менш продуктивним з точки зору швидкодії. Це особливо помітно в

обчислювально важких задачах, де швидкість виконання має велике значення. Крім того, Python має обмежену підтримку паралельного виконання, що може бути недоліком для деяких задач, які вимагають високої продуктивності і розподіленого обчислення.

Також, хоча Python має багато бібліотек і фреймворків, які полегшують розробку, вони не завжди мають однаково високу якість та стабільність. Деякі бібліотеки можуть бути менш популярними або менше підтримуватися спільнотою розробників, що може призвести до проблем з розширенням або підтримкою проекту.

Все ж таки, незважаючи на недоліки, Python залишається привабливою мовою програмування для моделювання небінарних кінцевих автоматів. Його простий синтаксис, багата екосистема та велика спільнота розробників роблять його потужним інструментом для реалізації складних автоматичних систем. Вибір мови програмування завжди залежить від конкретних потреб проекту та вмінь розробників, і Python може бути гарним варіантом для розробки небінарних кінцевих автоматів.

### **2.1.2 Мова гіпертекстової розмітки HTML**

HTML [11] (HyperText Markup Language) є стандартною мовою розмітки для створення веб-сторінок та веб-документів. Вона використовується для визначення структури та семантики контенту на веб-сторінках, таких як заголовки, абзаци, списки, таблиці, зображення та посилання.

Переваги HTML:

- простота в освоєнні: HTML має простий синтаксис, що дозволяє легко розуміти його структуру та використовувати базові теги для розмітки контенту;
- кросс-платформенність: HTML підтримується більшістю браузерів та пристроїв, що дозволяє забезпечити сумісність веб-сторінок на різних

платформах;

- розширюваність: HTML підтримує використання CSS (Cascading Style Sheets) для визначення зовнішнього вигляду сторінок;
- семантика: HTML надає багато вбудованих тегів, які дозволяють визначити семантику контенту, що полегшує пошукову оптимізацію та доступність веб-сторінок.

Недоліки HTML:

- обмежені можливості взаємодії з користувачем: HTML є мовою розмітки, і не забезпечує широких можливостей для обробки подій та взаємодії з користувачем. Для цього потрібно використовувати JavaScript або інші технології;
- обмежені можливості стилізації: HTML має базові можливості стилізації, але для складних та кастомних дизайнів часто потрібно використовувати CSS або інші засоби;
- відсутність динамічної зміни контенту: HTML використовується для статичного відображення контенту. Для динамічної зміни контенту, наприклад, оновлення даних без перезавантаження сторінки, необхідно використовувати JavaScript або AJAX-запити.

Всупереч своїм недолікам, HTML є ключовою мовою розмітки для веб-розробки, що надає структуру та семантику контенту. З його допомогою можна створювати ефективні та доступні веб-сторінки. Поєднання HTML з CSS і JavaScript дозволяє створювати багатофункціональні та візуально привабливі веб-додатки. Загальна простота та широке поширення HTML роблять його важливим інструментом для будь-якого веб-розробника.

### **2.1.3 Каскадна таблиця стилів-CSS**

CSS [12] (Cascading Style Sheets) - це мова стилів, що використовується для визначення зовнішнього вигляду веб-сторінок, написаних на мовах розмітки, таких як HTML та XML. Вона включає в себе

набір правил, які застосовуються до елементів сторінки, щоб визначити їх зовнішній вигляд та розташування.

#### Переваги CSS:

- розділення стилів та вмісту: CSS дозволяє чітко відокремити стилі від HTML-коду. Це спрощує редагування та підтримку веб-сторінок, оскільки зміни в стилях можна вносити безпосередньо в CSS-файли, не змінюючи структуру HTML;

- єдинообразний зовнішній вигляд: CSS дозволяє легко змінювати зовнішній вигляд всіх сторінок на сайті, вносячи зміни лише в один CSS-файл. Це забезпечує єдність стилю та полегшує підтримку та модернізацію веб-сайту;

- гнучкість та контроль: CSS надає широкі можливості для налаштування зовнішнього вигляду. Ви можете контролювати розміри, кольори, шрифти, відступи, рамки та інші аспекти елементів сторінки. Це дозволяє створювати унікальні та привабливі дизайни;

- підтримка адаптивного дизайну: CSS забезпечує можливість створювати адаптивні веб-сторінки, що пристосовуються до різних розмірів екранів. За допомогою медіа-запитів та розташування елементів можна змінювати відповідно до розміру пристрою, що дозволяє покращити взаємодію з користувачами на мобільних пристроях.

#### Недоліки CSS:

- складність: CSS може бути важко освоїти для початківців. Час від часу можуть виникати складнощі з розташуванням елементів, вирівнюванням та стилями;

- браузерна сумісність: Різні браузери можуть трактувати CSS-правила по-різному, що може призводити до некоректного відображення сторінок. Це вимагає тестування та додаткової роботи для забезпечення сумісності з різними браузерами;

- обмежена можливість обробки подій: CSS не надає повної можливості для обробки подій і взаємодії з користувачем. Для цього часто

потрібно використовувати JavaScript;

- складність управління стилями на великих проектах: На великих веб-сайтах з великою кількістю сторінок і елементів CSS може стати важким управляти. Збільшення розміру CSS-файлів та складність ієрархії стилів можуть призводити до збоїв у підтримці та обслуговуванні.

Незважаючи на ці недоліки, CSS залишається потужним інструментом для оформлення та стилізації веб-сторінок, і використовується всесвітньою спільнотою розробників для створення привабливих та функціональних інтерфейсів.

#### **2.1.4 PyGame Library**

PyGame [13] є відкритою бібліотекою для розробки ігор та графічних додатків у мові програмування Python. Вона надає набір інструментів для роботи з вікнами, зображеннями, звуком, анімацією та взаємодією з користувачем.

Бібліотека PyGame застосовується для розробки ігор, графічних симуляцій або просто як бібліотека для створення додатку з графічним інтерфейсом

Розробка ігор: Pygame використовується для створення 2D-ігор різних жанрів, включаючи аркади, платформери, головоломки, стратегії та інші.

Графічні симуляції: завдяки гнучкості та можливостям PyGame, його можна використовувати для створення графічних симуляцій, моделювання фізики, візуалізації даних та інших подібних проектів.

Додатки з графічним інтерфейсом: PyGame дозволяє розробляти додатки з графічним інтерфейсом, що спрощує створення інтерактивних програм з використанням зображень, кнопок, полів введення та інших елементів.

#### Переваги:

- простота використання: PyGame має простий та зрозумілий інтерфейс, що дозволяє швидко освоїти розробку ігор та графічних додатків;
- багатий функціонал: Бібліотека надає широкі можливості для роботи з зображеннями, звуком, анімацією, фізикою, обробкою подій та іншими аспектами розробки ігор;
- підтримка різних платформ: PyGame підтримує різні операційні системи, включаючи Windows, macOS та Linux, що робить його доступним для широкого кола розробників;
- інтеграція з Python: PyGame добре поєднується з мовою програмування Python, що є основною при написанні дипломної роботи.

#### Недоліки:

- обмежена 3D-підтримка: PyGame зорієнтований переважно на 2D-графіку, тому він може бути обмеженим у розробці складних 3D-ігор;
- відсутність вбудованих інструментів для роботи з фізикою: PyGame не має вбудованої підтримки для реалістичної симуляції фізики об'єктів, що вимагає використання додаткових бібліотек або власних реалізацій;
- низька продуктивність при великій кількості об'єктів: При обробці великої кількості об'єктів або виконанні складних обчислень PyGame може стикатися з питаннями продуктивності;
- відсутність активного розвитку: Останні оновлення PyGame не завжди відбуваються з високою частотою, що може призводити до відсутності нових функцій та покращень.

Незважаючи на недоліки, PyGame є потужною бібліотекою для розробки ігор та графічних додатків у мові програмування Python. З його допомогою було створено спрощену версію меню гри, для демонстрації моделі небінарного автомату.

## 2.2 Вибір середовища розробки

PyCharm [14] є одним з найпопулярніших і потужних інтегрованих середовищ розробки (IDE) для програмування мовою Python. Його доцільний вибір для розробки програм пов'язаний з безліччю переваг, які надає це середовище.

По-перше, PyCharm надає розширені можливості для підтримки роботи з Python. Воно підтримує велику кількість бібліотек і фреймворків. Беручи до уваги, що логіка проєкту була розроблена на мові програмування Python, PyCharm є вдалим вибором в якості середовища розробки.

По-друге, PyCharm має зручний та інтуїтивно зрозумілий інтерфейс, що сприяє продуктивній роботі розробників. Він надає широкі можливості для організації проєктів, керування залежностями, рефакторингу коду та інших процесів розробки.

По-третє, PyCharm підтримує інтеграцію з популярними інструментами розробки, такими як Git Docker. Це дозволяє розробникам працювати у знайомих середовищах та використовувати потужні інструменти для збереження та керування кодовою базою, що дає змогу розширювати проєкт.

Крім того, PyCharm пропонує інтегровану систему тестування, яка допомагає виявляти та виправляти помилки ще на ранніх етапах розробки.

Окрім розробки Python-програм, PyCharm також надає підтримку інших мов програмування, таких як HTML, CSS. Це дозволяє розробляти програму у веб середовищі, що стає зручним і підлягає розширенню у майбутньому, наприклад, захостити програму-сайт для більшої доступності серед користувачів.

Загалом, вибір PyCharm як середовища розробки для програмування мовою Python є доцільним, оскільки воно надає широкий функціонал, розширену підтримку Python та зручний інтерфейс.

### 2.3 Висновки до розділу 2

В цьому розділі було розглянуто основні інструменти для розробки програмного продукту, такі як основна мова програмування Python, HTML, CSS, бібліотека PyGame.

Було представлено загальну архітектуру програмного забезпечення, сформульовано функціональні вимоги до розроблюваного застосунку.

Також був наведений короткий опис текстового редактору PyCharm Community Edition Цей редактор має багато переваг, однією з яких є інтеграція основної мови розробки програми. З набором додаткових плагінів та бібліотек він стає ефективним інструментом для вирішення поставлених задач.

## 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 3.1 Опис алгоритму

Алгоритм виконання роботи моделі ліфта на основі FSM є наступним:

- ініціалізація: початковий стан моделі ліфта встановлюється як "спокійний" або "очікування". Усі необхідні змінні та параметри ініціалізуються;
- очікування на замовлення: ліфт перебуває в стані очікування на замовлення від пасажирів. Під час цього стану система очікує сигналів про натискання кнопок на поверхах будівлі або внутрішніх кнопок ліфта;
- прийняття замовлення: при отриманні замовлення ліфт переходить у стан "прийняття замовлення". В цьому стані визначається, на який поверх потрібно поїхати і які двері ліфта мають відкритися;
- рух до цільового поверху: ліфт переходить у стан "рух до цільового поверху". У цьому стані він рухається до поверху, на який надійшло замовлення. Двері ліфта залишаються закритими під час руху;
- зупинка на цільовому поверсі: при досягненні цільового поверху ліфт переходить у стан "зупинка на цільовому поверсі". Двері ліфта відкриваються, щоб пасажир міг увійти або вийти з ліфта;
- обробка зупинки: у стані "обробка зупинки" ліфт залишається на цільовому поверсі протягом заданого часу, щоб дозволити пасажирові виконати виходу або входу. Після цього ліфт переходить до наступного стану;
- повернення до стану очікування: після завершення обробки зупинки ліфт повертається до стану "очікування на замовлення" і очікує нового замовлення;
- перевірка наявності нових замовлень: у стані "очікування на замовлення" ліфт періодично перевіряє наявність нових замовлень. Якщо замовлення надходять, ліфт переходить до стану "прийняття замовлення" для обробки нового замовлення;

- завершення роботи: якщо не надійшло нових замовлень і ліфт перебуває в стані "очікування на замовлення" протягом певного періоду часу, робота моделі ліфта завершується.

Застосування FSM дозволяє ефективно керувати рухом ліфта, управляти дверима та обробляти замовлення в послідовних станах. Використання FSM для моделювання ліфта дозволяє забезпечити систему зрозумілим та легко, а також забезпечити гнучкість для розширення та модифікації функціональності ліфта. На рис. 3.1 надано загальний алгоритм переходу станів під час роботи моделі ліфта, модель якої побудована на основі небінарного кінцевого автомату.

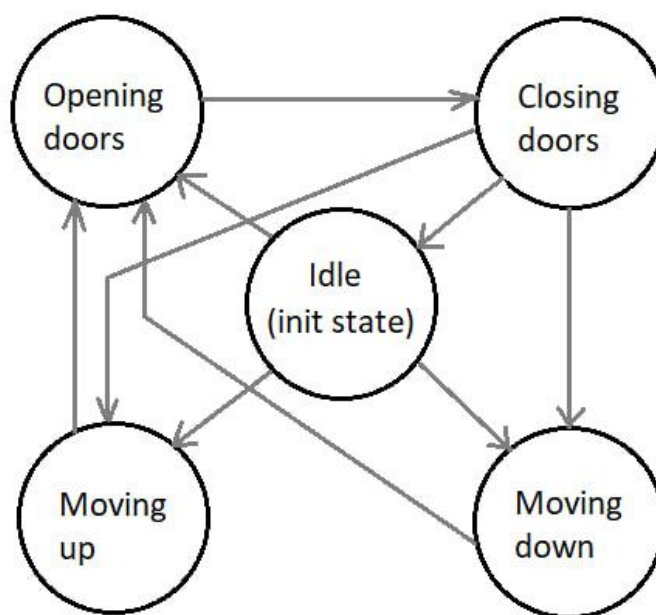


Рисунок 3.1 – Схема переходу станів ліфту на основі моделі небінарного автомату

Другим прикладом реалізації небінарних кінцевих автоматів є меню гри, що має наступний алгоритм:

- визначення станів: спочатку потрібно визначити список станів, які будуть присутні в меню. Наприклад, можуть бути стани «Меню-привітання», «Головне меню», «Меню паузи», «Геймплей»;
- створення класу стану: для кожного стану створюється окремий

клас, який успадковується від базового класу стану. У цьому класі визначаються функції для обробки подій, оновлення стану та відображення;

- ініціалізація станів: створюється об'єкт для кожного стану і встановлюються необхідні початкові значення;

- переходи між станами: в кожному стані визначаються умови, за яких відбуваються переходи до інших станів. Це можуть бути дії користувача, які спричиняють зміну стану (натискання кнопок, вибір пункту меню тощо);

- обробка подій: в кожному стані визначаються функції для обробки подій, які можуть виникати в цьому стані (натискання кнопок, рух миші тощо). Ці функції виконують відповідні дії і, за потреби, змінюють поточний стан або виконують перехід до іншого стану;

- оновлення стану: в кожному кадрі гри виконується оновлення поточного стану. Це може включати перевірку умов для переходу до інших станів або оновлення внутрішнього стану об'єктів, що знаходяться в цьому стані;

- відображення: в кожному кадрі гри виконується відображення поточного стану. Це може включати відображення тексту, графічних елементів, кнопок тощо;

- керування переходами: переходи між станами керовані за допомогою змінних, які зберігають поточний стан та наступний стан. Коли відбувається подія або умова для переходу, змінюється поточний стан на наступний, і виконується ініціалізація нового стану;

- головний цикл гри: в основному циклі гри перевіряється поточний стан і виконуються його функції обробки подій, оновлення та відображення. Цей цикл повторюється до тих пір, поки гра не закінчиться або користувач не змінить стан;

- вихід з гри: при виході з гри або закритті програми потрібно забезпечити коректну обробку і завершення програми.

### 3.2 Структурна схема проєкту

Програма розподілена на 2 блоки. Перший – модель описання ліфту, друга – модель описання ігрового меню. В цьому підрозділі буде описано структурну схему проєкту. Структура директорій та файлів проєкту представлена на рисунку 3.2.

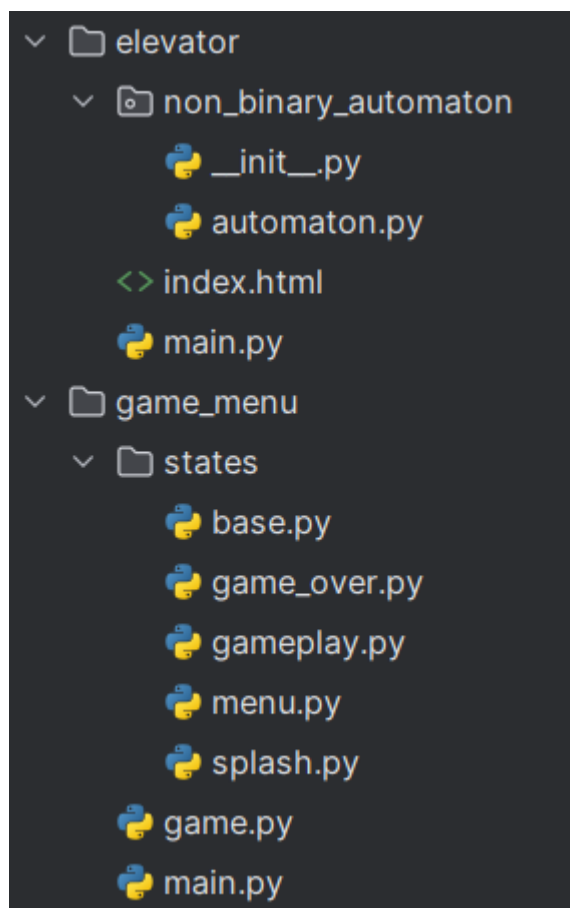


Рисунок 3.2 – Структура директорій та файлів проєкту

Директорія `elevator` призначена для зберігання всіх файлів, що стосуються опису моделі ліфту. У цій директорії знаходиться ще одна папка під назвою `non_binary_automaton` та інші файли.

Директорія `non_binary_automaton` містить в собі всі файли, які описують логіку роботи небінарного автомату ліфту.

Файл `index.html` містить в собі код для реалізації інтерфейсу програми.

Файл `main.py` у директорії `elevator`, містить в собі виклик основних

функцій роботи ліфту, призначення початкового стану.

Директорія `game_menu` призначена для зберігання усіх файлів, що стосуються опису моделі ігрового меню. У даній директорії міститься папка `states` з описом всіх можливих станів меню.

Окрім цього є також файли `main.py` та `game.py`. `Main.py`, який містить в собі підключення всіх станів з інших модулів разом з ініціалізацією усіх можливих станів. `Game.py` описує початкові властивості для гри.

### 3.3 Опис реалізованих функцій

В даному підрозділі будуть розглядатись деякі функції та змінні, які використовуються у проєкті. Розглянемо функції та змінні, що відповідають за реалізацію опису моделі небінарного автомата меню гри:

- `screen`: змінна для визначення розміру вікна;
- `states`: змінна словник для збереження набору можливих станів `MENU`, `SPLASH`, `GAMEPLAY`, `GAME_OVER`;
- `game`: звертається до модулю `game.py` до класу `Game`, описаний в ньому та передає розмір вікна (змінна `screen`), набір станів (змінна `states`) та початковий стан (`'SPLASH'`);
- `class Game`: цей клас має в собі опис початкових властивостей `done`, `screen`, `clock`, `fps`, `states`, `state_name`, `state`; та набір функцій `event_loop`, `flip_state`, `update`, `draw`, `run`;
- `done`: ця булева змінна використовується для перевірки чи потрібно завершувати гру. Гра завершується при значенні `done = True`;
- `screen`: змінна приймає передані значення з файлу `main.py` і містить в собі параметри розміру екрану застосунку;
- `Clock`, `fps`: змінні, які використовуються для визначення наскільки швидко буде рендеритися гра;

- `states`: змінна, яка зберігає в собі задані в файлі `main.py` значення, а саме словник із описаними станами;

- `state_name`: змінна, що зберігає в собі значення початкового стану;

- `event_loop()`: ця функція викликається у циклі гри та обробляє будь-які події, що відбуваються на даний момент часу. Наприклад, вихід із гри або натиснення кнопок;

- `flip_state()`: ця функція буде викликана при умові, що стан є виконаним і обробить перехід до наступного стану. Оновлює необхідні змінні, присвоює нові і передає будь-які постійні значення;

- `update()`: дана функція перевіряє чи потрібно виходити з гри, якщо нам потрібен перехід до нового стану, використовуючи попередню функцію `flip_state()`. Також ця функція буде викликана для обробки руху персонажа;

- `draw()`: функція, що надає команду намалювати екран, використовуючи посилання, які ми взяли раніше. Без цього нічого не буде відображатися. Також функція описує що буде малюватися на ігровому екрані у стані `gameplay`;

- `run()`: ця функція має основний цикл гри і буде працювати поки гри не буде надана команда вийти. Саме в цьому циклі обробляються будь-які події, які з'являються в `update`. По суті викликаються на виконання усі вищеописані функції;

- `class BaseState`: це основний клас, від якого йде логіка, а точніше наслідування для всіх інших станів. Окрім змінних, що налаштовують програму, цей клас має 4 функції: `startup`, `get_event`, `update`, `draw`.

При використанні моделі ліфта:

- `class ElevatorState`: цей клас є базовим класом для усіх станів ліфта. Він містить загальні методи та атрибути, які спільні для всіх станів. Кожен підклас цього класу буде відповідати конкретному стану ліфта і реалізувати

свою власну поведінку;

- class Stopped(): цей клас представляє стан "Зупинений". У цьому стані ліфт не рухається і очікує подій. Він обробляє подію up, яка запускає ліфт у рух, та оновлює поточний поверх та цільовий поверх;

- class MovingUp(): цей клас представляє стан "Рухається вгору". У цьому стані ліфт рухається вгору і очікує подій stop та down. Він також оновлює поточний поверх та встановлює новий цільовий поверх;

- class MovingDownState(LiftState): цей клас представляє стан "Рухається вниз". У цьому стані ліфт рухається вниз і очікує подій stop та up. Він також оновлює поточний поверх та встановлює новий цільовий поверх;

- class Lift: цей клас представляє модель ліфта. Він має властивість max\_floors (максимальна кількість поверхів, що дорівнює 5). Він містить метод change\_state, який змінює поточний стан ліфта на певний підклас ElevatorState, і метод handle\_event, який оброблює події, що стосуються руху ліфта.

Створюється об'єкт elevator класу Elevator з заданими параметром max\_floors.

За допомогою методу change\_state встановлюється початковий стан ліфта, Idle.

За допомогою методу handle\_event оброблюються події, наприклад ініціалізації початкових даних, шляхом взаємодії з інтерфейсом.

При отриманні події "Натиснуто кнопку на певному поверсі" (наприклад, event = "floor\_button\_pressed") викликається метод handle\_event об'єкта elevator. У методі handle\_event перевіряється поточний стан ліфта:

Якщо ліфт знаходиться у стані "Зупинений" (Stopped), встановлюється новий цільовий поверх, а потім викликається метод change\_state для переходу до стану "Рухається вгору" або "Рухається вниз" в залежності від

розташування цільового поверху.

Якщо ліфт знаходиться у стані "Рухається вгору" (MovingUp), перевіряється, чи знаходиться цільовий поверх нижче поточного. Якщо так, то встановлюється новий цільовий поверх. Якщо цільовий поверх вище поточного, подія не оброблюється.

Аналогічно, якщо ліфт знаходиться у стані "Рухається вниз" (MovingDown), перевіряється, чи знаходиться цільовий поверх вище поточного. Якщо так, то встановлюється новий цільовий поверх. Якщо цільовий поверх нижче поточного, подія не оброблюється.

При отриманні події "Зупинка ліфта" (наприклад, `event = "stop_button_pressed"`) викликається метод `handle_event` об'єкта `elevator`. У методі `handle_event` перевіряється поточний стан ліфта:

- якщо ліфт знаходиться у стані "Зупинений" (Idle), подія не оброблюється;

- якщо ліфт знаходиться у стані "Рухається вгору" (MovingUp) або "Рухається вниз" (MovingDownState), і в той же час викликається ліфт з іншого поверху – новий поверх ставиться у чергу викликів і чекає на виконання. Якщо напрям руху ліфту співпадає з новим заданою ціллю, ліфт може зупинитися на новому поверсі, перш ніж виконає попередню задачу.

### 3.4 Висновки до розділу 3

В даному розділі було детально описано алгоритм реалізації моделі кінцевих автоматів в супроводі зі схематичним графом, описуючим переходи станів роботи ліфту.

Разом з цим, були описані головні змінні, функції та класи, що виконують головну функцію – зміну станів на прикладі ігрового меню.

## **4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **4.1 Призначення програми**

Розроблювана програма призначена для створення та завантаження резюме, що в свою чергу допоможе ще на один крок приблизитися до успішного працевлаштування. Програмне забезпечення надає доступ до введення всієї необхідної інформації, а саме: персональні та контактні дані, супровідний лист, інтереси, рівень володіння тією чи іншою мовою, інформація про попередні місця роботи та розроблені проекти та ще багато якої інформації.

Програма дозволяє швидко та зручно створити якісне, сучасне резюме без зайвих зусиль та насолодитися процесом створення.

### **4.2 Умови виконання програми**

Для експлуатації розробленої програми необхідні наступні програмно-технічні засоби:

- комп'ютер з встановленою операційною системою Windows 11, 10, 8.1 або 7;
- процесор – не менше 800МГц;
- об'єм оперативної пам'яті пристрою – не менше 4 Гб для повноцінної та комфортної роботи;
- монітор та відеокарта з роздільною здатністю не менше 1920x1080, а також смартфон;
- обсяг вільного простору на HDD чи SSD не менше 1 Гб;
- підключення до мережі Інтернет.

### 4.3 Тестування в критичних середовищах

Метою даного тестування є перевірка програмного продукту на стійкість та коректність роботи у критичних ситуаціях. Тестування включає в себе перевірку на велику кількість одночасних вхідних сигналів, довготривала робота, ввід непередбачуваних даних, наявність стійкості та не відмови від обслуговування при неправильній взаємодії користувача з програмним продуктом.

Для тестування було взято операційні системи Windows 11, 10, та 7, різний виділений об'єм оперативної пам'яті та певну кількість виділених ядер на процес. Для отримання більш точних результатів тестування проводилося на жорсткому диску (HDD). Має підключення до Інтернет.

Тестування надало наступні результати:

- велика кількість одночасних вхідних сигналів: система піддається тестуванню з великою кількістю одночасно активних вхідних сигналів. Це допомагає перевірити, чи впорається система з великим навантаженням та правильно обробляє всі вхідні сигнали без збоїв чи втрати даних. Тестування пройдено успішно;

- довготривала робота: система піддається тестуванню на тривалість роботи в критичних умовах протягом тривалого періоду часу. Це допомагає перевірити стійкість системи, забезпечити, щоб не виникали проблеми через довготривалу роботу. Тестування пройдено успішно;

- введення непередбачуваних даних: під час тестування з введенням непередбачуваних даних можна визначити, як система реагує на такі ситуації. Це включає виявлення та обробку помилок, забезпечення правильної реакції системи на некоректні дані, виключення можливості виникнення конфліктів або небажаних станів, а також забезпечення стабільності та безпеки системи при обробці непередбачуваних даних.. Тестування пройдено успішно.

Варто зауважити, що робота програми є неможливою за умов відсутності під'єднання до мережі Інтернет.

#### 4.4 Тестування програми

Для розробленого ПЗ було проведено повноцінне тестування: перевірка відповідності текстів заданій темі, перевірка відповідності до індивідуального завдання, перевірка на адаптивність, перевірка на зрозумілість та ясність інтерфейсу, перевірка кнопок та коректність відгуку програми на введені користувачем дані.

Після перевірки заповнення шаблону інформацією були отримані такі результати: виявлено відсутність орфографічних, стилістичних та пунктуаційних помилок, тексти відповідають заданій тематиці, а також успішно пройдена мовна перевірка. Результат перевірки – без помилок.

Під час перевірки придатності до застосування, виявлено, що програма має інтуїтивно зрозумілий інтерфейс.

Були проведені стрес-тести на виявлення можливих помилок та збоїв в роботі програмного забезпечення. Результат тестування – коректне відображення вивідних даних та, в загальній складності, роботи продукту. Результати роботи можна побачити на рис. 4.1 - 4.3.

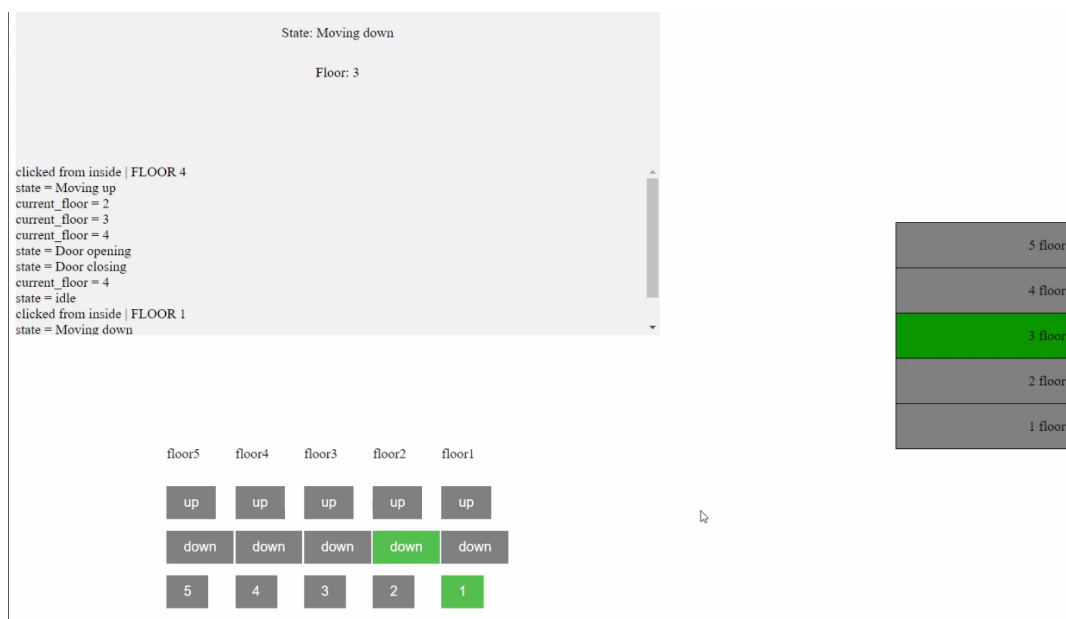


Рисунок 4.1 – Коректна робота моделі роботи ліфту

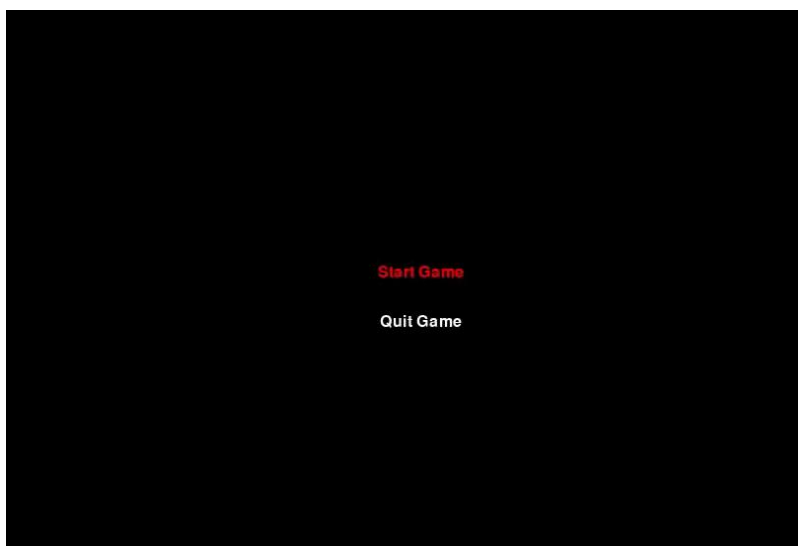


Рисунок 4.2 – Коректна робота моделі ігрового меню (головне меню)



Рисунок 4.3 – Коректна робота моделі ігрового меню (пауза)

#### **4.5 Висновки до розділу 4**

В даному розділі було проведено повноцінне тестування програми для моделювання небінарних автоматів. Під час тестування було встановлено, що програма має інтуїтивно зрозумілий та зручний інтерфейс. Також були проведені випробування в критичних умовах, що показало, що програма може працювати в будь-яких умовах. Після завершення тестів можна зробити висновок, що програмне забезпечення працює коректно та відповідає всім вимогам, що були поставлені.

## ВИСНОВКИ

У результаті дослідження та розробки програми для моделювання небінарних автоматів було досягнуто важливих результатів. Програма дозволяє ефективно моделювати поведінку небінарних автоматів, що має велике значення в різних областях.

Програма має інтуїтивно зрозумілий інтерфейс, що спрощує її використання навіть користувачам з мінімальним досвідом. Вона підтримує 2 види небінарних автоматів та забезпечує доступ до наочної демонстрації різних сценаріїв.

Однією з найбільших переваг цієї програми є точна працездатність, що було доказано шляхом проведення мануального тестування в критичних середовищах. Крім того, програма успішно пройшла тестування, під час якого було перевірено її функціональність та надійність. Виявлено відсутність помилок та проблем з орфографією, стилістикою та пунктуацією.

Загалом, розроблена програма для моделювання небінарних автоматів є потужним інструментом, який дозволяє демонструвати роботу небінарних автоматів з високою точністю та надійністю. Вона може бути використана у навчальних цілях, допомагаючи розширити розуміння та використання небінарних автоматів у різних сферах.

Крім того, розроблена програма спрощує роботу з небінарними автоматами завдяки своїм зручним інструментам та можливостям. Вона надає можливість візуалізувати стани та переходи автомата, аналізувати його поведінку, а також зберігати та завантажувати моделі для подальшого використання. Це робить програму ефективним інструментом для моделювання та вивчення небінарних автоматів.

Висновки з цієї дипломної роботи підтверджують, що розроблена програма для моделювання небінарних автоматів є цінним інструментом для дослідження та проектування систем, що використовують небінарні автомати. Вона допомагає покращити розуміння та аналіз цих систем, спрощує їх розробку та тестування в різних сценаріях. Розроблена програма має потенціал для подальшого розвитку та застосування в різних областях, включаючи інформаційні технології, автоматизацію та штучний інтелект.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Анякін, В. Скінченний автомат [Електрон. ресурс] / В. Анякін. – Режим доступу: <http://surl.li/icooy>
2. MATLAB – Understanding State Machines, Part 1: What Are They? [Electronic resource]. – Access mode: [https://www.youtube.com/watch?v=Yicg2TTMPs&ab\\_channel=MATLAB](https://www.youtube.com/watch?v=Yicg2TTMPs&ab_channel=MATLAB)
3. Поляков, М.О. Кінцеві автомати з небінарними елементами множин / М.О. Поляков, І.А. Андріас // Системні технології : регіональний міжвузівський збірник наукових праць. – 2018. – № 3 (116). – С. 85–90. ISSN 1562-9945
4. Кисляньська, Д.В. Моделі часових керуючих автоматів з гнучкою логікою на основі мікроконтролерів [Електрон. ресурс] / Д.В. Кисляньська. – Режим доступу: <https://openarchive.nure.ua/server/api/core/bitstreams/b75d893e-f90d-437e-a7fa-2cb8b74a6014/content>
5. Sipser, M. Introduction to the Theory of Computation / M. Sipser. – 3rd ed.: Cengage Learning International, 2012. – С. 31-39. ISBN-13: 978-1-133-18779-0S
6. Goldi-labs.net [Electronic resource]. – Access mode: <https://www.goldi-labs.net/>
7. MATLAB [Electronic resource]. – Access mode: <https://uk.mathworks.com/products/matlab.html>
8. MATLAB – Computer calculations in education, science and engineering [Electronic resource]. – Access mode: [https://www.researchgate.net/profile/Serhiy-Shtovba/publication/332655611\\_MATLAB\\_and\\_computer\\_calculations\\_in\\_education\\_science\\_and\\_engineering/links/5cc2068792851c8d2204ed97/MATLAB-and-](https://www.researchgate.net/profile/Serhiy-Shtovba/publication/332655611_MATLAB_and_computer_calculations_in_education_science_and_engineering/links/5cc2068792851c8d2204ed97/MATLAB-and-)

computer-calculations-in-education-science-and-engineering.pdf

9. SkyCAD Electrical [Electronic resource]. – Access mode: <https://skycad.ca/>

10. Coursera – What Is Python Used For – [Electronic resource]. – Access mode: <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>

11. MDN web docs – HTML: HyperText Markup Language [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/HTML>

12. MDN web docs – CSS: Cascading Style Sheets [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/CSS>

13. PyGame library [Electronic resource]. – Access mode: <https://www.pygame.org/docs/>

14. PyCharm – How PyCharm Can Help You Code Faster and Better Using Python [Electronic resource]. <https://emeritus.org/blog/coding-what-is-pycharm/>

**ДОДАТОК А**  
**Технічне завдання**

## **A.1 Підстави для розробки**

Підставою для розробки програмного забезпечення служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Програма для моделювання небінарних автоматів", затверджене наказом № 87 від 04 квітня 2023 р. по НУ "Запорізька політехніка".

## **A.2 Призначення розробки**

Дане програмне забезпечення призначене для демонстрації роботи моделі кінцевих автоматів на прикладі ліфту та меню гри. Керування ліфтом користувач може здійснювати самостійно, шляхом взаємодії з інтерфейсом. Модель ігрового меню представлено схематично і демонструє переходи станів у різних етапах гри.

## **A.3 Вимоги до програми**

### **A.3.1 Вимоги до функціональних характеристик**

Розроблюване програмне забезпечення має надавати користувачу наступні можливості:

- моделювання небінарних автоматів;
- симуляція роботи автоматів;
- вивід проміжних даних для більш детального розуміння роботи FSM;
- інтуїтивно зрозумілий інтерфейс користувача;
- можливість обирати модель для демонстрації роботи FSM;
- оброблювач на некоректно введених вхідні параметрів;
- можливість для розширюваності.

Вхідними даними, необхідними для функціонування програмного продукту, є взаємодія користувача з інтерфейсом, що спровокує обробку запитів користувача для симуляції небінарних автоматів. Вихідними даними

є робота небінарного автомату, що відображається в інтерфейсі та проміжні значення, станів, які виводяться на екран протягом роботи програми.

### **А.3.2 Вимоги до надійності**

Розроблюване програмне забезпечення повинно не допускати помилку у виконанні розробленого алгоритму, шляхом непередбачуваної взаємодії з інтерфейсом програми. Наприклад, при натисненні на іншу кнопку, що не є передбачуваною для задуманого користування програмою.

### **А.3.3 Умови експлуатації**

Умови експлуатації повинні відповідати вимогам, які пред'являються до технічних засобів. Показники температури, відносної вологості і швидкості руху повітря повинні відповідати оптимальним і допустимим значенням в робочій зоні виробничих приміщень.

### **А.3.4 Вимоги до складу та параметрів технічних засобів**

Серед технічних засобів мають бути наявні:

- ОС Windows 7 та вище;
- ОЗУ 2 Гб або більше;
- наявність вільного місця на жорсткому диску більше 5 Гб;
- наявність клавіатури, миші, монітору.

### **А.3.5 Вимоги до інформаційної та програмної сумісності**

Розроблюване ПЗ повинно підтримувати спільний режим роботи з ОС Windows. Програма повинна бути підготовлена до можливої модернізації. Для реалізації програмного продукту необхідно використовувати мову

програмування Python та середовище розробки PyCharm Community.

Інформаційні структури, використані на вході та на виході, мають відповідати визначеним структурам даних.

### **A.3.6 Вимоги до програмної документації**

Включно із розроблюваним програмним забезпеченням повинно бути розроблено керівництва програміста та оператора, а також опис програми.

### **A.4 Стадії та етапи розробки**

Розробка даного програмного продукту повинна включати наступні етапи:

- дослідження та аналіз теорії моделі небінарних кінцевих автоматів;
- проєктування й опис архітектури програми;
- розробка програми;
- тестування програми.

### **A.5 Порядок приймання і тестування**

Тестування даного програмного забезпечення проводиться у вигляді виконання усіх можливих сценаріїв оператора.

Дане програмне забезпечення вважається прийнятним, якщо воно задовольняє усім висунутим вимогам до нього і успішно пройшло етап тестування.

**ДОДАТОК Б**  
**Текст програми**

## Б.1 Файл main.py

```
import sys
import pygame
from game_menu.states.menu import Menu
from game_menu.states.gameplay import Gameplay
from game_menu.states.game_over import GameOver
from game_menu.states.splash import Splash
from game import Game

pygame.init()
screen = pygame.display.set_mode((1920, 1080))
states = {
    "MENU": Menu(),
    "SPLASH": Splash(),
    "GAMEPLAY": Gameplay(),
    "GAME_OVER": GameOver(),
}

game = Game(screen, states, "SPLASH")
game.run()

pygame.quit()
sys.exit()
```

## Б.2 Файл game.py

```
import pygame

class Game(object):
    def __init__(self, screen, states, start_state):
        self.done = False
        self.screen = screen
        self.clock = pygame.time.Clock()
        self.fps = 60
        self.states = states
        self.state_name = start_state
        self.state = self.states[self.state_name]
```

```

def event_loop(self):
    for event in pygame.event.get():
        self.state.get_event(event)

def flip_state(self):
    current_state = self.state_name
    next_state = self.state.next_state
    self.state.done = False
    self.state_name = next_state
    persistent = self.state.persist
    self.state = self.states[self.state_name]
    self.state.startup(persistent)

def update(self, dt):
    if self.state.quit:
        self.done = True
    elif self.state.done:
        self.flip_state()
    self.state.update(dt)

def draw(self):
    self.state.draw(self.screen)

def run(self):
    while not self.done:
        dt = self.clock.tick(self.fps)
        self.event_loop()
        self.update(dt)
        self.draw()
        pygame.display.update()

```

### Б.3 Файл base.py

```

import pygame

class Game(object):
    def __init__(self, screen, states, start_state):
        self.done = False

```

```

self.screen = screen
self.clock = pygame.time.Clock()
self.fps = 60
self.states = states
self.state_name = start_state
self.state = self.states[self.state_name]

def event_loop(self):
    for event in pygame.event.get():
        self.state.get_event(event)

def flip_state(self):
    current_state = self.state_name
    next_state = self.state.next_state
    self.state.done = False
    self.state_name = next_state
    persistent = self.state.persist
    self.state = self.states[self.state_name]
    self.state.startup(persistent)

def update(self, dt):
    if self.state.quit:
        self.done = True
    elif self.state.done:
        self.flip_state()
    self.state.update(dt)

def draw(self):
    self.state.draw(self.screen)

def run(self):
    while not self.done:
        dt = self.clock.tick(self.fps)
        self.event_loop()
        self.update(dt)
        self.draw()
        pygame.display.update()

```

## Б.4 Файл `game_over.py`

```

import pygame
from .base import BaseState

class GameOver(BaseState):
    def __init__(self):
        super(GameOver, self).__init__()
        self.title = self.font.render("Game Over", True, pygame.Color("white"))
        self.title_rect = self.title.get_rect(center=self.screen_rect.center)
        self.instructions = self.font.render("Press space to start again, or enter
to go to the menu", True, pygame.Color("white"))
        instructions_center = (self.screen_rect.center[0],
self.screen_rect.center[1] + 50)
        self.instructions_rect =
self.instructions.get_rect(center=instructions_center)

    def get_event(self, event):
        if event.type == pygame.QUIT:
            self.quit = True
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_RETURN:
                self.next_state = "MENU"
                self.done = True
            elif event.key == pygame.K_SPACE:
                self.next_state = "GAMEPLAY"
                self.done = True
            elif event.key == pygame.K_ESCAPE:
                self.quit = True

    def draw(self, surface):
        surface.fill(pygame.Color("black"))
        surface.blit(self.title, self.title_rect)
        surface.blit(self.instructions, self.instructions_rect)

```

## Б.5 Файл `gameplay.py`

```
import pygame

from .base import BaseState


class Gameplay(BaseState):

    def __init__(self):
        super(Gameplay, self).__init__()

        self.rect = pygame.Rect((0, 0), (80, 80))

        self.rect.center = self.screen_rect.center

        self.next_state = "GAME_OVER"

    def get_event(self, event):

        if event.type == pygame.QUIT:

            self.quit = True

        elif event.type == pygame.KEYUP:

            if event.key == pygame.K_UP:

                self.rect.move_ip(0, -10)

            if event.key == pygame.K_DOWN:

                self.rect.move_ip(0, 10)

            if event.key == pygame.K_LEFT:

                self.rect.move_ip(-10, 0)

            if event.key == pygame.K_RIGHT:

                self.rect.move_ip(10, 0)

            if event.key == pygame.K_SPACE:

                self.done = True

    def draw(self, surface):

        surface.fill(pygame.Color("black"))

        pygame.draw.rect(surface, pygame.Color("blue"), self.rect)
```

## Б.6 Файл menu.py

```

import pygame
from .base import BaseState

class Menu(BaseState):
    def __init__(self):
        super(Menu, self).__init__()
        self.active_index = 0
        self.options = ["Start Game", "Quit Game"]
        self.next_state = "GAMEPLAY"

    def render_text(self, index):
        color = pygame.Color("red") if index == self.active_index else
pygame.Color("white")
        return self.font.render(self.options[index], True, color)

    def get_text_position(self, text, index):
        center = (self.screen_rect.center[0], self.screen_rect.center[1] + (index *
50))

        return text.get_rect(center=center)

    def handle_action(self):
        if self.active_index == 0:
            self.done = True
        elif self.active_index == 1:
            self.quit = True

    def get_event(self, event):
        if event.type == pygame.QUIT:
            self.quit = True
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_UP:
                self.active_index = 1 if self.active_index <= 0 else 0
            elif event.key == pygame.K_DOWN:
                self.active_index = 0 if self.active_index >= 1 else 1
            elif event.key == pygame.K_RETURN:
                self.handle_action()

```

```

def draw(self, surface):
    surface.fill(pygame.Color("black"))
    for index, option in enumerate(self.options):
        text_renderer = self.render_text(index)

        surface.blit(text_renderer,
self.get_text_position(text_renderer, index))

```

## Б.7 Файл splash.py

```

import pygame
from .base import BaseState

class Splash(BaseState):
    def __init__(self):
        super(Splash, self).__init__()
        self.title = self.font.render("Incompetent Game", True,
pygame.Color("blue"))
        self.title_rect = self.title.get_rect(center=self.screen_rect.center)
        self.next_state = "MENU"
        self.time_active = 0

    def update(self, dt):
        self.time_active += dt
        if self.time_active >= 5000:
            self.done = True

    def draw(self, surface):
        surface.fill(pygame.Color("black"))
        surface.blit(self.title, self.title_rect)

```

## Б.8 Файл main.py

```

import uvicorn
from fastapi import FastAPI
from starlette.websockets import WebSocket

from elevator import Elevator

```

```

app = FastAPI()
elevator = None

@app.get("/call_elevator/{floor}")
def call_elevator(floor: int):
    elevator.call_from_floor(floor)
    return {"message": f"Elevator called from floor {floor}"}

@app.get("/call_from_inside/{floor}")
async def read_item(floor: int):
    if elevator is None:
        return 'refresh the page'

    print(floor)
    await elevator.call_from_inside(floor)

@app.get("/call_from_outside/{floor}/{go_up}")
async def read_item(floor: int, go_up: bool):
    if elevator is None:
        return 'refresh the page'
    print(floor, go_up)
    await elevator.call_from_outside(floor, go_up)

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    global elevator

    elevator = Elevator(websocket=websocket)
    await websocket.send_json({'success': True})
    while True:
        data = await websocket.receive_json()
        print(data['op'])

        if data['op'] == 'call_from_inside':
            floor = data['floor']
            await elevator.call_from_inside(floor)
        elif data['op'] == 'call_from_outside':

```

```

        floor = data['floor']
        go_up = data['go_up']
        await elevator.call_from_outside(floor, go_up)

if __name__ == '__main__':
    uvicorn.run(app, host='localhost', port=8089)

```

## Б.9 Файл index.html

```

<!DOCTYPE html>
<html>
<head>
  <title>WebSocket Example</title>
  <style>
    /* CSS for the button and layout */
    .container {
      display: flex;
      height: 100vh;
    }

    .left {
      flex: 1;
      display: flex;
      flex-direction: column;
    }

    .top {
      flex: 1;
      display: flex;
      flex-direction: column;
    }

    .top-left {
      flex: 1;
      background-color: #f1f1f1;
      display: flex;
      flex-direction: column;
      align-items: center;

```

```

    justify-content: center;
}

.top-right {
    flex: 1;
    /*display: flex;*/
    /*justify-content: center;*/
    /*align-items: center;*/
    background-color: #f1f1f1;
    overflow: auto;
    max-height: 200px
}

current_floor,
current_state {
    margin-bottom: 10px; /* Add some bottom margin for spacing */
}

.bottom {
    flex: 1;
    display: flex;
    align-items: center;
    justify-content: center;
}

/* CSS for the button */
.elevator-button {
    background-color: gray; /* Gray by default */
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin-top: 10px;
    cursor: pointer;
}

.elevator-button.clicked {
    background-color: #4CAF50; /* Green when clicked */
}

```

```

    }

    .right {
        flex: 1;
        display: flex;
        justify-content: center;
        align-items: center;
    }
</style>
<script type="text/javascript">

    function sendHttpRequest(url) {
        var xhttp = new XMLHttpRequest();
        xhttp.open("GET", url, true);
        xhttp.send();
        console.log("Sent HTTP request to:", url);
    }

    function sendCallFromOutside(from_floor, go_up) {
        if (!document.getElementById(`${from_floor}floor${go_up ? 'up' :
'down'}`).classList.contains("clicked")) {
            var url = `http://localhost:8089/call_from_outside/${from_floor}/${go_up}`;
            sendHttpRequest(url);
            // Change button color to green when clicked
            document.getElementById(`${from_floor}floor${go_up ? 'up' :
'down'}`).classList.add('clicked');
            if (go_up) {
                document.getElementById("logs").innerHTML += `clicked from outside |
FLOOR ${from_floor} | GO UP<br>`
            } else {
                document.getElementById("logs").innerHTML += `clicked from outside |
FLOOR ${from_floor} | GO DOWN<br>`
            }
        }
    }

    function sendElevatorCall(floor) {
        if
(!document.getElementById(`${floor}floor`).classList.contains("clicked")) {
            var url = `http://localhost:8089/call_from_inside/${floor}`;
            sendHttpRequest(url);
            // Change button color to green when clicked

```

```

        document.getElementById(`${floor}floor`).classList.add('clicked');
        document.getElementById("logs").innerHTML += `clicked from inside | FLOOR
${floor}<br>`
    }
}

// Establish WebSocket connection
var socket = new WebSocket("ws://localhost:8089/ws");

// WebSocket event: connection established
socket.onopen = function(event) {
    console.log("WebSocket connection established");
};

// WebSocket event: message received
socket.onmessage = function(event) {
    var message = JSON.parse(event.data);
    console.log("Received message:", message);
    var lastLog = ""
    var logsElement = document.getElementById("logs");
    var logsContent = logsElement.innerHTML;
    var logsArray = logsContent.split("<br>");
    if (logsArray.length > 1) {
        lastLog = logsArray[logsArray.length - 2];
    }

    current_log = `${message.op} = ${message[message.op]}`
    if (lastLog !== current_log && message.op) {
        document.getElementById("logs").innerHTML += `${current_log}<br>`
    }
    if (message.op === "state") {
        document.getElementById("current_state").innerHTML = `State:
${message.state}`;

        if (message.state === 'Door opening') {
            var doors = document.querySelectorAll(".door");
            var current_floor = 1
            for (var door in doors) {
                current_floor ++;
                if (doors.hasOwnProperty(door)) {
                    if (doors[door].parentElement.style.backgroundColor === "green") {

```

```

        current_floor =
parseInt(document.getElementById("current_floor").innerText.split("Floor: ")[1])
        console.log(current_floor, "current_floor")

document.getElementById(`${current_floor}floorup`).classList.remove('clicked');

document.getElementById(`${current_floor}floordown`).classList.remove('clicked');

document.getElementById(`${current_floor}floor`).classList.remove('clicked');

        openDoor(doors[door].parentElement);
    }
}
}
}
if (message.state === 'Door closing') {
    var doors = document.querySelectorAll(".door");
    for (var door in doors) {
        if (doors.hasOwnProperty(door)) {
            console.log(doors[door].parentElement.style.backgroundColor);
            if (doors[door].parentElement.style.backgroundColor === "green") {
                closeDoor(doors[door].parentElement);
            }
        }
    }
}
}

if (message.op === "current_floor") {
    document.getElementById("current_floor").innerHTML = `Floor:
${message.current_floor}`;
    for (let i = 1; i <= 5; i++) {
        document.getElementById(`${i}door`).parentElement.style.backgroundColor =
"gray";
    }

document.getElementById(`${message.current_floor}door`).parentElement.style.backgroundCo
lor = "green";
}
};

```

```

// WebSocket event: connection closed
socket.onclose = function(event) {
    console.log("WebSocket connection closed");
};

</script>
</head>
<body>
    <div class="container">
        <div class="left">
            <div class="top">
                <div class="top-left">
                    <a id="current_state">State: idle</a><br>
                    <a id="current_floor">Floor: 1</a><br>
                </div>
                <div id="logs" class="top-right">
                </div>
            </div>
            <div class="bottom">
                <table>
                    <tr>
                        <td><a>floor5</a>
                        <td><a>floor4</a>
                        <td><a>floor3</a>
                        <td><a>floor2</a>
                        <td><a>floor1</a>
                    </tr>
                    <tr>
                        <td><button id="5floorup" class="elevator-button"
onclick="sendCallFromOutside(5, true)">up</button></td>
                        <td><button id="4floorup" class="elevator-button"
onclick="sendCallFromOutside(4, true)">up</button></td>
                        <td><button id="3floorup" class="elevator-button"
onclick="sendCallFromOutside(3, true)">up</button></td>
                        <td><button id="2floorup" class="elevator-button"
onclick="sendCallFromOutside(2, true)">up</button></td>
                        <td><button id="1floorup" class="elevator-button"
onclick="sendCallFromOutside(1, true)">up</button></td>
                    </tr>
                    <tr>
                        <td><button id="5floordown" class="elevator-button"

```

```

onclick="sendCallFromOutside(5, false)">down</button></td>
        <td><button id="4floordown" class="elevator-button"
onclick="sendCallFromOutside(4, false)">down</button></td>
        <td><button id="3floordown" class="elevator-button"
onclick="sendCallFromOutside(3, false)">down</button></td>
        <td><button id="2floordown" class="elevator-button"
onclick="sendCallFromOutside(2, false)">down</button></td>
        <td><button id="1floordown" class="elevator-button"
onclick="sendCallFromOutside(1, false)">down</button></td>
    </tr>
<tr>
        <td><button id="5floor" class="elevator-button"
onclick="sendElevatorCall(5)">5</button></td>
        <td><button id="4floor" class="elevator-button"
onclick="sendElevatorCall(4)">4</button></td>
        <td><button id="3floor" class="elevator-button"
onclick="sendElevatorCall(3)">3</button></td>
        <td><button id="2floor" class="elevator-button"
onclick="sendElevatorCall(2)">2</button></td>
        <td><button id="1floor" class="elevator-button"
onclick="sendElevatorCall(1)">1</button></td>
    </tr>
</table>
</div>
</div>
<div class="right">
    <div id="imageContainer">

<style>
table {
    border-collapse: collapse;
    width: 200px;
}

td {
    height: 50px;
    width: 50px;
    position: relative;
}

```

```

.door {
  position: absolute;
  top: 0;
  left: 0;
  width: 0;
  height: 100%;
  background-color: white;
  transition: width 0.5s;
  z-index: 1;
}

td.open .door {
  width: 50%;
}
</style>
<script>
  function animateDoor(element) {
    element.classList.toggle("open");
  }

  function openDoor(doorElement) {
    // Add the class to open the door
    doorElement.classList.add("open");
  }

  function closeDoor(doorElement) {
    // Remove the class to close the door
    doorElement.classList.remove("open");
  }

</script>
<table>
  <tr>
    <td style="border: 1px solid black; background-color: gray; text-align:
right;">
      <div id="5door" class="door"></div>
      5 floor
    </td>
  </tr>
</table>

```

```

        <td style="border: 1px solid black; background-color: gray; text-align:
right;">
            <div id="4door" class="door"></div>
            4 floor
        </td>
    </tr>
    <tr>
        <td style="border: 1px solid black; background-color: gray; text-align:
right;">
            <div id="3door" class="door"></div>
            3 floor
        </td>
    </tr>
    <tr>
        <td style="border: 1px solid black; background-color: gray; text-align:
right;">
            <div id="2door" class="door"></div>
            2 floor
        </td>
    </tr>
    <tr>
        <td style="border: 1px solid black; background-color: green; text-align:
right;">
            <div id="1door" class="door"></div>
            1 floor
        </td>
    </tr>
</table>
</div>
</div>
</div>
</body>
</html>

```

## Б.10 Файл \_\_init\_\_.py

```
from .elevator import Elevator
```

## Б.11 Файл `automaton.py`

```
import asyncio
from typing import Union

class Elevator:
    FLOORS = 5
    DEFAULT_FLOOR = 1

    def __init__(self, websocket=None):
        self._current_floor = 1
        self._busy = False
        self._passengers_amount = 0
        self._passengers_weight = 0
        self._doors_opened = False
        self._queue_from_inside = []
        self._queue_from_outside = set()
        self._going_to = None

        self._websocket = websocket
        self._state = None
        self._going_up = False

    async def need_to_stop(self):
        if self._current_floor == self._going_to:
            try:
                self._queue_from_outside.remove((self._current_floor, self._going_up))
            except KeyError:
                ...
            try:
                self._queue_from_outside.remove((self._current_floor, not self._going_up))
            except KeyError:
                ...

        elif (self._current_floor, self._going_up) in self._queue_from_outside:
            await self.open_close_doors()

    if self._going_up:
        await self._websocket.send_json({'op': 'state', 'state': 'Moving up'})
    else:
```

```

        await self._websocket.send_json({'op': 'state', 'state': 'Moving down'})
        await self._websocket.send_json({'op': 'current_floor', 'current_floor':
self._current_floor})
        try:
            self._queue_from_outside.remove((self._current_floor,
self._going_up))
        except KeyError:
            ...
        try:
            self._queue_from_outside.remove((self._current_floor, not
self._going_up))
        except KeyError:
            ...

    async def call_from_outside(self, from_floor: Union[str, int], go_up: bool,
to_print=True):

        from_floor = int(from_floor)
        if to_print:
            await self._websocket.send_json({'op': 'called_from_floor',
'called_from_floor': from_floor})

        if self._busy and to_print:
            self._queue_from_outside.add((from_floor, go_up))
            return
        self._busy = True

        if from_floor > self._current_floor:
            self._state = 'up'
            self._going_up = True
            await self._websocket.send_json({'op': 'state', 'state': 'Moving up'})

        elif self._current_floor > from_floor:
            self._state = 'down'
            self._going_up = False
            await self._websocket.send_json({'op': 'state', 'state': 'Moving
down'})

        await self._websocket.send_json(f'going from {self._current_floor} to
{from_floor}')

```

```

self._going_to = from_floor
await self.change_floor(from_floor - self._current_floor)
self._busy = False

    if self._queue_from_inside:
        await self.call_from_inside(self._queue_from_inside.pop(),
to_print=False)

    if self._queue_from_outside:
        value = self._queue_from_outside.pop()
        await self.call_from_outside(value[0], value[1], to_print=False)

    await self._websocket.send_json({'op': 'state', 'state': 'idle'})

    async def call_from_inside(self, to_floor: Union[str, int], to_print=True):
        to_floor = int(to_floor)
        if to_print:
            await self._websocket.send_json({'op': 'called_to_floor',
'called_to_floor': to_floor})

    if self._busy and to_print:
        self._queue_from_inside.append(to_floor)
        return

self._busy = True

    if to_floor > self._current_floor:
        self._state = 'up'
        self._going_up = True
        await self._websocket.send_json({'op': 'state', 'state': 'Moving up'})
    elif self._current_floor > to_floor:
        self._state = 'down'
        self._going_up = False

    await self._websocket.send_json({'op': 'state', 'state': 'Moving down'})
    await self._websocket.send_json(f'going from {self._current_floor} to
{from_floor}')
    self._going_to = to_floor

    await self.change_floor(to_floor - self._current_floor)
    self._busy = False

```

```

        if self._queue_from_inside:
            await self.call_from_inside(self._queue_from_inside.pop(),
to_print=False)

        if self._queue_from_outside:
            value = self._queue_from_outside.pop()
            await self.call_from_outside(value[0], value[1], to_print=False)

        await self._websocket.send_json({'op': 'state', 'state': 'idle'})

    async def open_close_doors(self):
        if not self._doors_opened:
            self._state = 'door_opening'
            await asyncio.sleep(0.5)
            await self._websocket.send_json({'op': 'state', 'state': 'Door
opening'})

            await asyncio.sleep(1)
            await self._swap_doors()
            self._state = 'door_closing'
            await self._websocket.send_json({'op': 'state', 'state': 'Door
closing'})

            await asyncio.sleep(1)
            await self._swap_doors()
        async def go_up(self):
            self._current_floor += 1
        async def go_down(self):
            self._current_floor -= 1

    async def change_floor(self, times: int):
        if times < 0:
            self.open_close_doors()
            for _ in range(abs(times)):
                await asyncio.sleep(1.5)
                await self.go_down()
                await self._websocket.send_json({'op': 'current_floor',
'current_floor': self._current_floor})
                await self.need_to_stop()

            elif times > 0:
                self.open_close_doors()

```

```

        for _ in range(times):

            await asyncio.sleep(1.5)
            await self.go_up()
                await self._websocket.send_json({'op': 'current_floor',
'current_floor': self._current_floor})
                await self.need_to_stop()
                await self.open_close_doors()
                self._state = None

                await self._websocket.send_json({'op': 'current_floor',
'current_floor': self._current_floor})
            async def _swap_doors(self):
                self._doors_opened = not self._doors_opened

```

**ДОДАТОК В**  
**Слайди презентації**

НУ «Запорізька політехніка»  
Кафедра програмних засобів  
Дипломна кваліфікаційна робота бакалавра

## Програма для моделювання небінарних автоматів

Виконала  
ст. групи КНТ-210сп

Єлизавета ШИКІНА

Керіник  
д.т.н., професор

Михайло ПОЛЯКОВ

Рисунок В.1 – Слайд 1

### Інформація про проєкт

- **Метою** даного проєкту є розробка програмного продукту, що матиме можливість продемонструвати роботу небінарного кінцевого автомату
- **Об'єкт** дослідження – є самі небінарні автомати. В рамках проєкту проводиться дослідження їх властивостей, особливостей та можливих застосувань.
- **Предмет** дослідження – способи опису моделі небінарних автоматів та реалізація програми.

Рисунок В.2 – Слайд 2

# Задачі

1. Проаналізувати предметну область.
2. Розробити технічне завдання.
3. Спроектувати програмне забезпечення.
4. Розробити програмне забезпечення.
5. Оформити документацію.

Рисунок В.3 – Слайд 3

## Аналіз предметної області

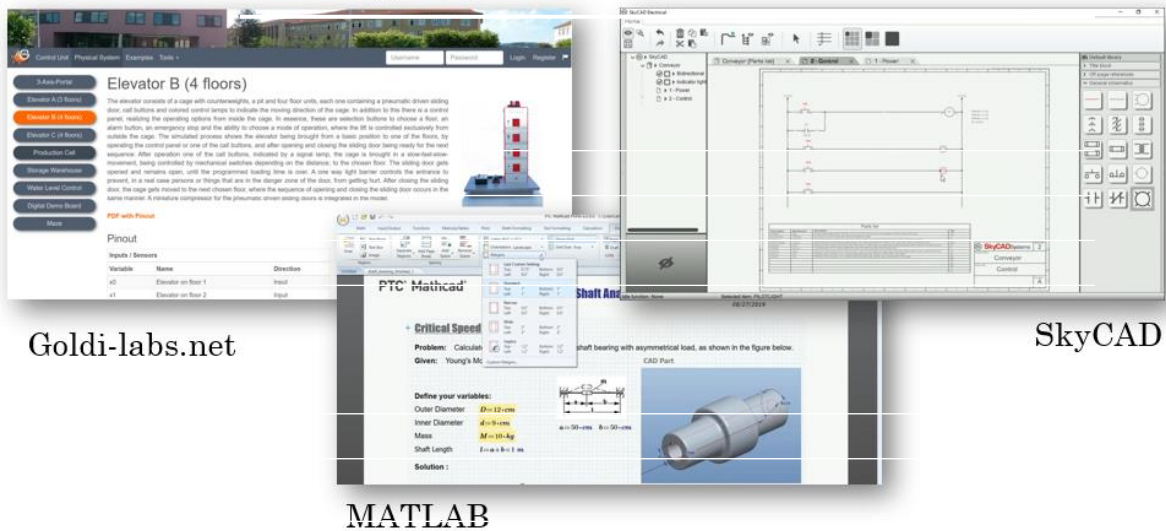


Рисунок В.4 – Слайд 4

## Вибір мови для розробки

|                               | Java | C++ | Python |
|-------------------------------|------|-----|--------|
| Легкість у вивченні           | +·   | -   | +      |
| Екосистема                    | ++   | +   | ++     |
| Швидкодія                     | +++  | +++ | ++     |
| Можливість розширення         | ++   | +++ | +++    |
| Компільована / інтерпретована | K    | K   | I      |
| Портативність                 | ++   | +   | ++     |
| Швидкість розробки            | +    | +   | +++    |

Рисунок В.5 – Слайд 5

## Розробка алгоритму

- Вузли: Idle (очікування), Moving Up (рух вгору), Moving Down (рух вниз), Closing doors (зачинення дверей), Opening doors (відчинення дверей).
- Ребра: натискання кнопки у середині ліфту або ззовні на поверсі; таймаут очікування.

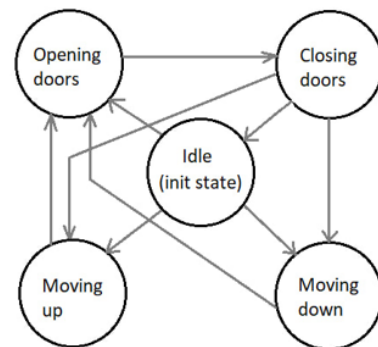


Рисунок В.6 – Слайд 6

## Реалізація

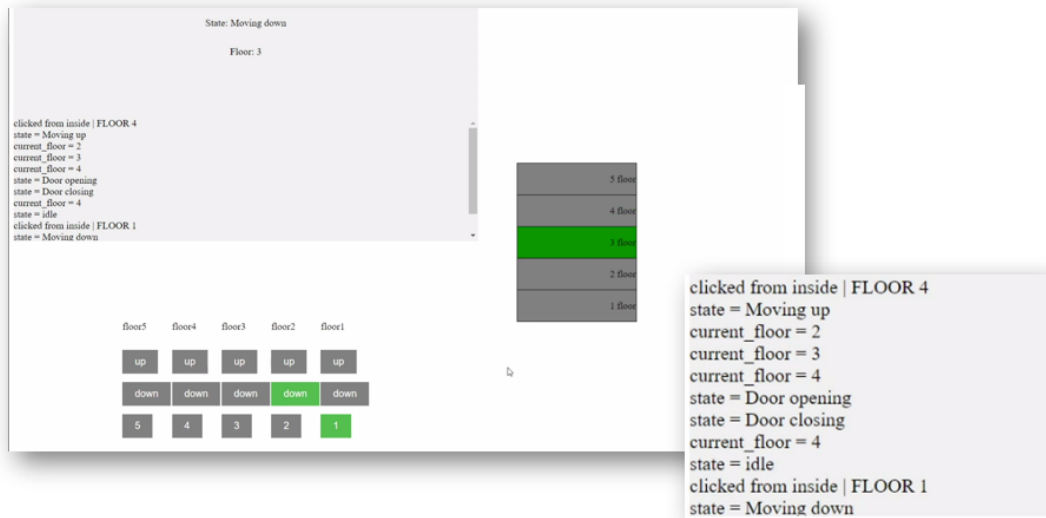


Рисунок В.7 – Слайд 7

## Реалізація

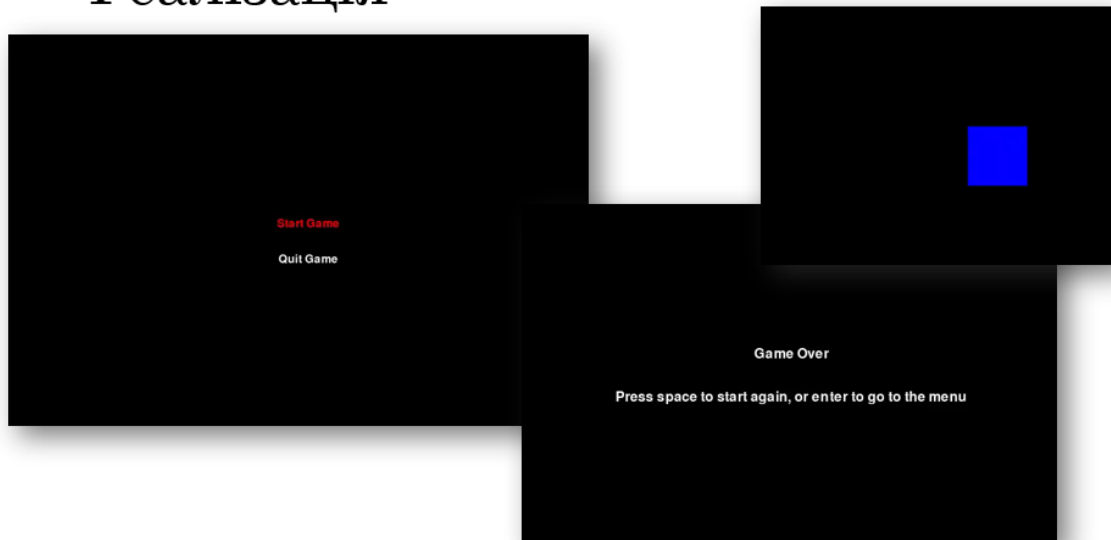


Рисунок В.8 – Слайд 8

## Висновки

- В ході виконання дипломної роботи було досліджено предметну область та розроблено програмне забезпечення яке моделює роботу небінарних автоматів.
- Розроблена система відповідає поставленим задачам, програмної сумісності на є надійним протестованим продуктом.
- На практиці дане ПЗ може використовуватися у навчальних цілях, допомагаючи розширити розуміння та використання небінарних автоматів у різних сферах.

Рисунок В.9 – Слайд 9

Дякую за увагу!

Рисунок В.10 – Слайд 10