2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
Paper ID #51618

Design and Programming of an Elevator Simulator for Teaching Programmable Logic Controller Course

Dr. Wenle Zhang, University of Arkansas at Little Rock

Dr. Wenle Zhang is an Associate Professor of Electrical and Electronics Program in the Dept. of Engineering Technology of University of Arkansas at Little Rock since 2007. He received his B.S. and M.S. from Shandong Univ. of Sci. and Tech. in 1983 and Shanghai Univ. of Sci. and Tech. in 1986, respectively. He received his Ph.D. in Electrical Engineering from Ohio University in 2000. Prior to pursuing his Ph.D. at Ohio University, he was employed with Rockwell Automation, a famous industrial automation supplier, as a control engineer for more than 5 years. He has experience in PLC application, industrial control networking, control software development. Since spring of 2001, Dr. Zhang has been an assistant professor in the School of Electrical Engineering and Computer Science at Ohio University. Dr. Zhang's current research interests include discrete event system, industrial automation, system identification and neural networks.

Design and Programming of an Elevator Simulator for Teaching Programmable Logic Controllers Course

Wenle Zhang
School of Engineering and Engineering Technology
University of Arkansas at Little Rock

A. Introduction

To help our students' learning in the Programmable Logic Controllers (PLCs) classes, zPLC – a PLC system simulator software was designed since 2020 during the COVID19, so that students can practice their programming skills both on and off campus with zPLC, which consists of a soft PLC, a basic I/O device console simulator (presented at ASEE 2024), see Fig 1. Built on top of zPLC, a 4-floor elevator simulation software, the elevator simulator, was designed to mimic the mini elevator in our PLC lab (Fig 2., it was broken at the time) while waiting for it to be repaired. The goal was to build a simple yet fully functional virtual 4-floor elevator with similar functions provided by the mini elevator, listed below:

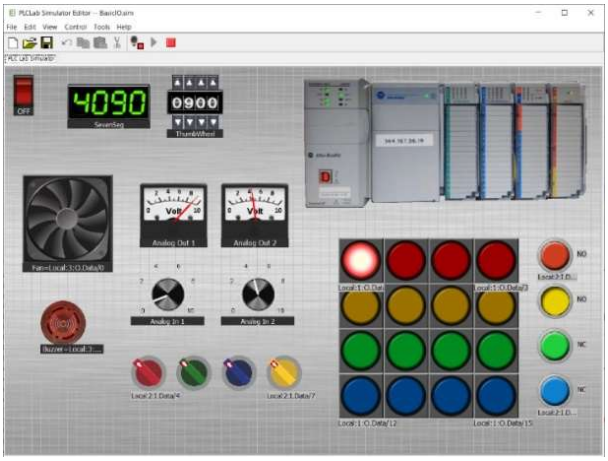

Fig 1. zPLC - The PLC System Simulator

- Fully working model of an elevator with four floors
- Floor sensing and visual indication
- Motorized elevator car with direction control of travel
- Motorized elevator car door open, close and bell
- 'Normally on' brake to hold car at a desired floor
- Up/down call buttons on each floor and inside car

The elevator simulator should provide a realistic application to fully illustrate the principles of PLC I/O interfacing and control so that our students could practice programming skill on it. The elevator simulator was designed with an object model-based method. To be consistent with the GUI of zPLC, the elevator simulator was designed to have a GUI with a look and feel familiar to users of most recent Windows-based software, as shown in Fig 3. The resulting elevator simulator was first used for the elevator control programming project in the summer 2020 Robotics and Programmable Logic Controllers class. Programming of the simulator with zPLC was based on a state transition diagram (or state machine) design method which was taught in that class.

B. Design and Functions of the Elevator Simulator

Architecture and functions of the elevator simulator were designed to have a realistic appearance of a 4-floor building with push buttons, indicators, floor sensors, motorized car with brake and motorized door, all are equipped with animated visual effects, including floor sensing illumination, inside light in car once

door opened, with blue background as seen in Fig 2 and 3, and more importantly with I/O addresses (or tags) for zPLC integration, as seen in Fig 3. All these virtual device components of the elevator simulator should be accessible and controllable by zPLC. The designed operational functions included the normal functions of an elevator system, and the elevator control I/O interface for zPLC integration.

Fig 2. The mini elevator

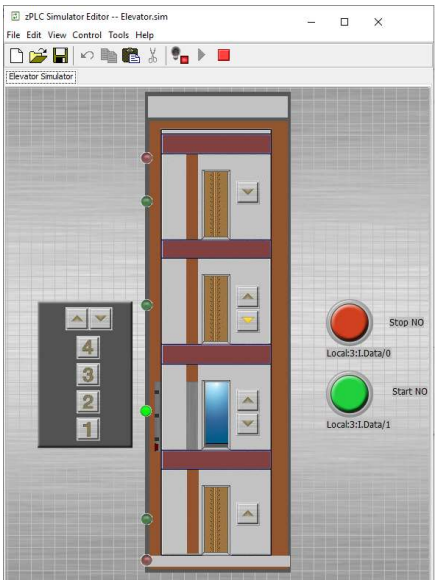

Fig 3. Architecture of the elevator simulator

B.1 Normal Functions of an Elevator System

The elevator car moves between floors in response to requests from call buttons on each floor and call buttons on the control panel inside the elevator car (shown as an external panel left of the elevator). Sensors indicate which floor the elevator car is at and provide advance warning to allow the car to be slowed prior to arriving at the destination floor. A brake allows the car to be held on a floor while the motorized car door is operated to allow occupants to enter or exit and a bell is provided to announce arrival and to warn door opening/closing. The elevator simulator provides visually animated car speed control with logic controlled by zPLC with minimal digital I/O. Car position feedback provided by the 4-floor sensors. These position feedback signals allow advanced control of the elevator car motion while approaching a floor to first slow down speed then stop with braking.

B.2 I/O Interface Sub-System for zPLC Integration

Shown in Fig 4, the I/O interface sub-system was formed by many components including 4 floor sensors, 6 on-floor and 4 in-car call buttons for destination floor all with indicator lights, and the door opened and closed sensors. 2-speed elevator motion control, direction control, brake control, door open and close, and bell control. For zPLC integration, almost all components are assigned with input or output addresses (or simply I/O addresses or I/O tags). Sensors and push buttons are assigned input addresses accessible to and read by zPLC, while lights, speed and brake and direction of car motion, car bell and door open and closing, are assigned with output addresses accessible and written from or controlled by zPLC.

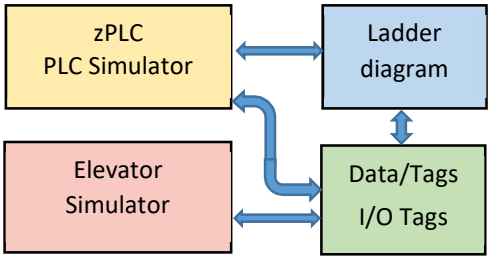

Fig 4. zPLC integration to the elevator simulator

Input addresses and corresponding sensors are summarized in Table 1 and output addresses for controllable components are summarized in Table 2.

Integration of the elevator simulator to zPLC through the I/O interface sub-system is the process of configuration in which all animated components on the simulator are configured to have matching I/O addresses which will be controlled by the ladder diagram program running in zPLC.

Table 1. Input addresses

Input address	Description
IN0, IN1, …, IN5	Floor call pushbuttons: F1-Up, F2-Dn, F2-Up, F3-Dn, F3-Up, F4-Dn
IN6, IN7, IN8, IN9	Pushbuttons on control pad inside car: F1-P, F2-P, F3-P, F4-P
IN10, IN11,IN12, IN13	Floor number sensors: F1-S, F2-S, F3-S, F4-S
IN14, IN15	Door open/close sensors: D-Opened, D-Closed
IN16, IN17	Start/Stop pushbutton

Table 2. Output addresses

Output address	Description
OUT0, OUT1, …, OUT5	Illuminate Floor LED: L1-Up, L2-Dn, L2-Up, L3-Dn, L3-Up, L4-Dn
OUT6, OUT7, OUT8, OUT9	Illuminate Floor LED on control pad inside car: L1-P, L2-P, L3-P, L4-P
OUT10	Open/close car door: Door
OUT11	Release/apply car brake: Brake
OUT12, OUT13, OUT14	Car speed 30% On/Off, 70% On/Off, Car direction Up/Down: Sp30, Sp70, Dir
OUT15	Bell On/Off to announce arrival of car at the floor: Bell
OUT16, OUT17	Car travel direction LED on control pad inside car: Car-Up, Car-Dn

B.3 PLC Controlled Elevator Operation

With the elevator simulator integration to zPLC completed, the elevator control application program can be written up in Ladder Diagram language, refer to Allen-Bradley manuals and references for the ControlLogix series PLCs [1-5], IEC 61131 document [7]. Once zPLC is entered with a correct elevator application program, both the elevator and zPLC can be placed in RUN mode and the elevator should operate normally. One such operation scenario can be like:

- The car is parked on first floor initially
- Press either the button on 1st floor or button 1 inside car should open the car door, ring the bell, delay a few seconds then close.
- If any other floor buttons (irrespective of up/down) pushed, the car should move up to that floor, slow down, stop and apply brake, ring the bell, open the door, delay a few seconds and then close.

A screenshot of zPLC running the elevator simulator control application program is shown in Fig 5. The elevator control application program can be entered into zPLC by placing zPLC in Edit mode – when the program is being edited with the ladder diagram editor of zPLC, a routine (a program unit) can be entered, modified and saved. Once the entire working program is entered completely, zPLC can be placed in RUN mode, the program can no longer be edited, and zPLC now functions as a live ladder diagram monitor and is animated with live data fed from the elevator simulator, see Fig 5.

B.4. Components Design of the Elevator Simulator

In addition to the appearance of the building, the elevator simulator consists of numerous device components: a total number of 14 push buttons, 6 sensors (4 for 4 floor sensors, 2 for hard limits top/bottom), 19 LED lights (12 of them integrated into the call buttons, 4 for 4 floor sensors, 2 for hard limits top/bottom, 1 for brake), 4 doors (each has 2 sensors for door being opened and closed) and the elevator car with 3

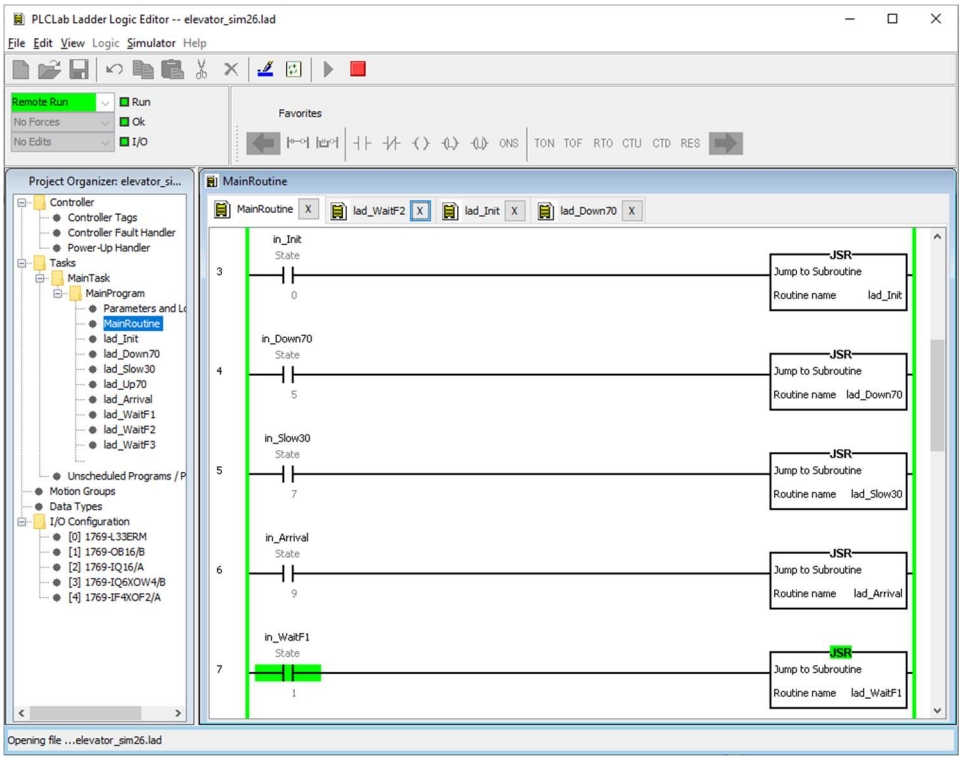

Fig 5. Main routine of elevator program in zPLC

prongs for triggering floor sensor 3 times at each floor moving up or down and a brake. Basic devices, such as a push button, a limit switch and an indicator light, are defined as components. They are used to form larger devices (or equipment) of different types, such as the in-car panel as a device with multiple buttons and light mounted, the car door as a device with a motor for door opening/closing and 2 limit switches indicating the door being opened or closed. Therefore, to design the elevator simulator with all these devices, a model-based method with UML and object oriented design were adopted [6] [11].

There will be only one base object for all type of components needs to be modeled, as the BaseComp in Fig 6. BaseComp will have multiple *attributes*/properties, such as, component *name*, *description*, *type*, *style*, *size* and *I/O address array*, etc. BaseComp is the parent object to all actual components and every actual component would be some sort of extension to the base object, with extra attributes, such as, *state*,

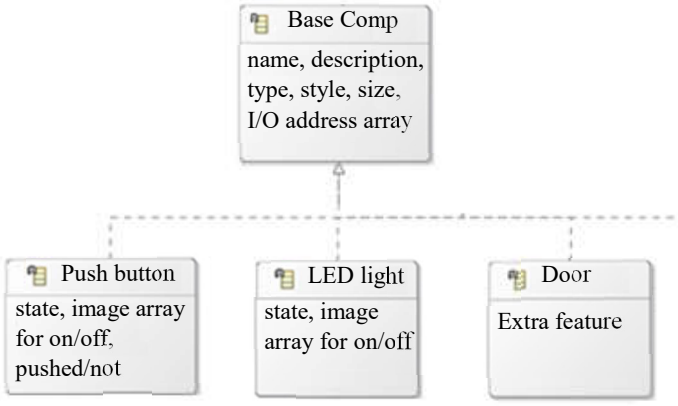

Fig 5. Object Model by UML

color or *image* and *position arrays* for different component states given from or to one or more I/O addresses in the I/O address array, and etc.

For example, an illuminated push button as an extended object to BaseComp with extra attributes, maybe be built as:

name: F1-Up
description: go up button on first floor with LED light
type: push button with light
style: square
size: 32 x 32 pixels
I/O address array: IN5(Push button states), OUT3 (for arrow LED light, externally triggered)
Extra attributes:
push button state: 0 – not pushed (use Off image), 1 – being pushed (use On image)
LED light state: 0 – off (use Off image), 1 – on (use On image)
Image array: pushed/On image  , pushed/Off image , normal/On image normal/Off image .

B.5 Design implementation

The designed elevator simulator was implemented in Java – an object-oriented programming language, since it was designed with an object model-based method and with major components being heavily GUI-based. With the model-based design, programming and testing was an intuitive and straightforward process and the resulting 1st version elevator simulator was a success. To be consistent with the GUI of zPLC, the elevator simulator was designed to have a GUI with a look and feel familiar to users of most recent Windows-based software, as shown in Fig 3.

C. Teaching PLC Programming of the Elevator Simulator

The elevator simulator was first used for the elevator control programming project in the summer 2020 Robotics and Programmable Logic Controllers class, then also in fall 2022 and 2024. Programming of the simulator with zPLC was based on a state transition diagram design method which was taught in the class. The elevator simulator provides a realistic PLC control application for the PLC class in which the students were assigned a programming project to control the elevator on the simulator with zPLC. zPLC with the elevator simulator were installed on both the lab computers and individual students' computers so that they could practice their PLC programming skills both on campus and at home. However, the project was not an easy task for the students, because this realistic application is quite sophisticated. To solve this issue, the students were taught to first design a state transition diagram and then convert to the PLC control program in Ladder Diagram language. The development of the state transition diagram for the elevator was an instructor lead process in which students were required to identify some of states and to fill in the details about behavior of their states. For example, the instructor would illustrate state *Down_70* and state *Wait_F2* with what needed to be done in each state and the students would identify state *Up_70*, *Wait_F1*, *Wait_F3* and *Wait_F4* and figure out what needed to be done in each state. Students were also required to determine most transition events with a few transitions provided by the instructor.

C.1 State Transition Diagram

A state machine or a state transition diagram is commonly defined[11] as: a graphic diagram that describes the logical transition of a system through various states of operation by representing states as ovals or rectangles, the transitions as directed lines/arcs that connect states, and the events that trigger transitions. The state transition diagram for the elevator control was designed as shown in Fig 6. Where states and transitions (correspondingly triggering events) are identified (refer to Tables 1 & 2).

C.2 States for the Elevator Operation

There are a total of nine states were identified and the behavior of each state is described here:

Init – initialization of the elevator control system
Down_70 – elevator moving down at normal speed, Sp70 = ON, Dir = OFF (down, see down arrow in Fig 7.), Brake = ON (release, as red LED see Fig 7.)
Up_70 – elevator moving up at normal speed, Sp70 = ON, Dir = ON (up, OFF for down), Brake = ON (release)
Slow_30 – elevator moving at a slow speed, Sp30 = ON, Sp70 = OFF
Arrival – elevator arrives at destination floor, Sp30 = OFF, Brake = OFF (applied), Bell = ON (for 3 sec), Door = ON (open) for 5 sec, then OFF
Wait F1, Wait F2, Wait F3, Wait F4 -- wait at the floor and wait for a call button pressed

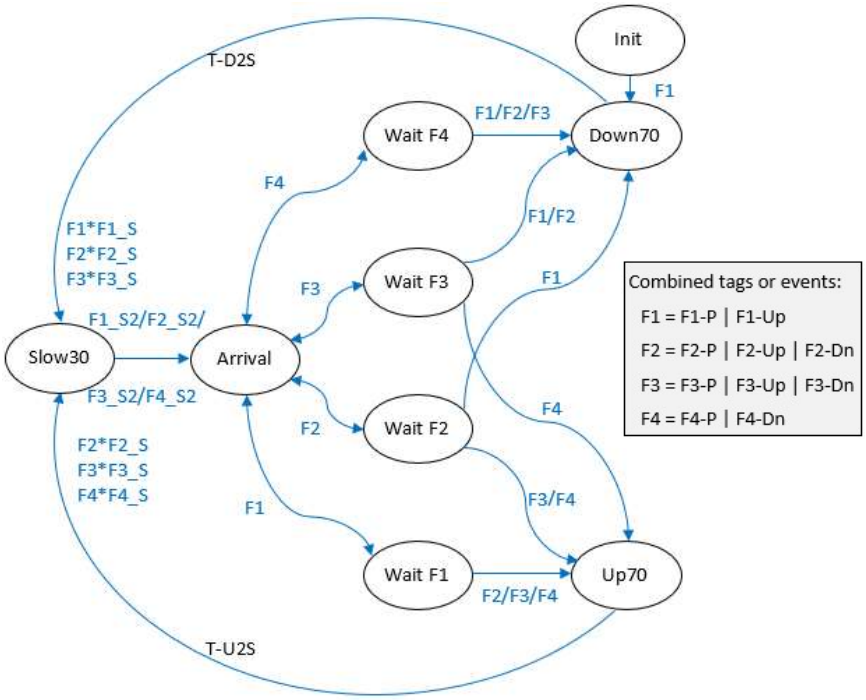

Fig 6. State transition diagram of the Elevator Simulator control program

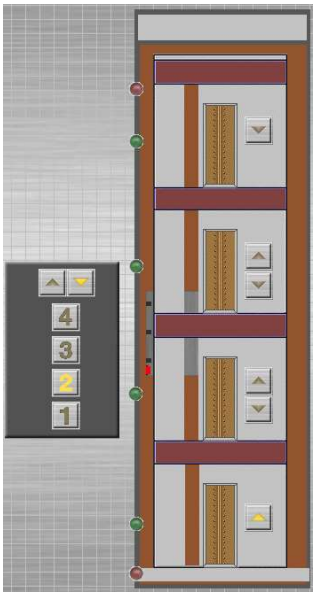

Fig 7. Animated call buttons, direction and brake

C.3 Transitions for the Elevator Operation

There are a total eighteen transition among states were identified and they are listed below with triggering events:

T-I2D – from Init to Down70, F1 (see combined tags or events in the Fig 6.)
T-D2S – from Down70 to Slow30, F1 * F1-S | F2 * F2-S | F3 * F3-S, while going to floor 1 to 3 with the corresponding floor sensor triggered.
T-U2S – ditto but for floor 2 to 4.
T-S2A – from Slow30 to Arrival, F1-S2 | F2-S2 | F3-S2 | F4-S2, floor 1 sensor is triggered 2nd time or floor 2 sensor is, etc.
T-A2W1/*T-W12A* – from Arrival to Wait F1, F1 (going to floor 1) / from Wait F1 to Arrival, F1 (call from floor 1)
T-A2W2/*T-W22A* – ditto for floor 2
T-A2W3/*T-W32A* – ditto for floor 3
T-A2W4/*T-W42A* – ditto for floor 4
T-W12U – from Wait F1 to Up70, F2 | F3 | F4, call from floor 2, 3 or 4
T-W22U – from Wait F2 to Up70, F3 | F4, call from floor 3 or 4
T-W32U – from Wait F3 to Up70, F4, call from floor 4
T-W22D – from Wait F2 to Down70, F1, call from floor 1
T-W32D – from Wait F3 to Down70, F1 | F2, call from floor 1 or 2
T-W42D – from Wait F4 to Down70, F1 | F2 | F3, call from floor 1, 2 or 3

C.4 State Transition Diagram Conversion to PLC Program

With the state transition diagram given, conversion to PLC control program in Ladder Diagram language can be performed as described below. Refer to manuals and references for the ControlLogix series PLCs [1-5] of Allen-Bradley.

- First, assign a BOOL tag/bit to each state, such as *in_Init* for the initial state *Init*, *in_Down70* for state *Down70*, etc.
- Second, converting a state: create a routine for every state that holds the ladder logic for the state activities, where routine name has prefix *lad_*. E.g., *lad_Init* is the routine name for state *Init*, *lad_Down70* is for state *Down70*.
- Third, converting a transition: upon a transition being set to ON (by the triggering events), system will leave or exit the from-state of transition (unlatch OFF the state tag as first action in the state routine) and will enter the to-state of transition (latch ON the state tag as last action in the state routine). E.g., If *T-I2D* (triggered by F1, from *Init* to *Down70*) is ON, then *in_Init* is reset or unlatched OFF first in routine *lad_Init* and *in_Down70* is set or latched ON last in routine *lad_Init*. See Fig 8.
- Then, in the main ladder program, make a subroutine call to a state's routine if the system is in that state. Where *in_Init* is preset (latched to ON), so system always starts in state *Init*. Part of the *MainRoutine* is shown Fig 5.

C.5 Using the Elevator Simulator in PLC Classes

Since the first version of this elevator simulator was used in our 2020 summer Robotics and PLC course and was evaluated [9], it has been gradually improved and used every other year alternating between the elevator simulator (latest fall 2024) and the lab mini work cell production system (fall 2025). With the use of zPLC/elevator simulator, we were able to make our summer 2020 (during COVID19) Robotics and Programmable Logic Controllers course offering a complete online class, in which each student worked on

the elevator PLC programing project and got an installation of zPLC/elevator simulator on their own home computer. Subsequently, the elevator PLC programing project were assigned in classes offered as in-person during fall 2022 and fall 2024.

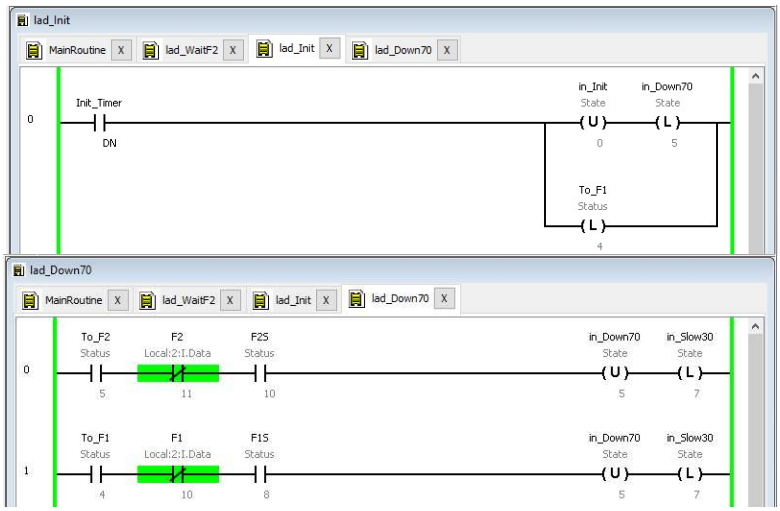

Fig 8. lad_Init and lad_Down70 routines of elevator program in zPLC

D. Conclusions and Future Work

The goal was set to build a simple yet fully functional software elevator simulator with similar functions provided by the mini elevator, the proposed elevator simulation software integrated with zPLC – the PLC simulator, was designed and implemented in Java. And the result is considered as a "Mission Accomplished". The elevator simulator together with zPLC software can be used for online PLC course offerings and for students' lab assignments at home. Future work includes developing a simulator for the mini work cell production system which assembles a metal/plastic washer onto a metal base peg. The work cell will consist of 12 input sensors: a start and a stop push button, 6 photo sensors, a height Ok sensor, height top/bottom dead center sensors and an inductive sensor for sorting plastic/metal washer, and 10 output actuators: a height measurement motor, 2 conveyor motors, 2 parts dispensers (solenoids), 2 sorting chutes (solenoids) and 3 sorting flippers (solenoids). On the upper conveyor belt, it performs washer height detection (Ok if thickness is 8mm, bad if it's 7mm/9mm), inductive detection for metal/plastic washers, sorting washers of metal/plastic to one chute or the other, and scraping washers of bad height to end of belt bin. Assembling of washer to base peg will be done on the lower conveyor with sorting at the end of belt.

References

[1] ControlLogix System User's Manual, Rockwell Automation Publication 1756-UM001P-EN-P - May 2017
[2] Logix5000 Controllers General Instructions Reference Manual, Rockwell Automation Publication 1756-RM003T-EN-P - November2018
[3] Logix 5000 Controllers I/O and Tag Data, Rockwell Automation Publication 1756-PM004H-EN-P - Feb 2018
[4] Logix 5000 Controllers Ladder Diagram, Rockwell Automation Publication 1756-PM008G-EN-P - Feb 2018
[5] Logix 5000 Controllers Tasks, Programs, and Routines, Rockwell Automation Publication 1756-PM005H-EN-P - February 2018
[6] Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User's Guide. Addison-Wesley, Reading, Mass., 1999.
[7] IEC 61131-3:2013, Programmable controllers - Part 3: Programming languages. 3rd ed. Geneva: International Electrotechnical Commission, 2013.
[8] Java reference: https://docs.oracle.com/javase/8/docs/api/overview-summary.html
[9] Wenle Zhang. Design of a PLC System Simulator and Application to Teaching Programmable Logic Controller Course Online. ASEE 2024 Annual Conference, Jun 2024, Portland, Oregon
[10] Feedback Instruments Limited, FBK Elevator 34-150 manual, Crowborough, England, Feb 2007
[11] UML 2 Tutorial - State Machine Diagram, https://sparxsystems.com/resources/tutorials/uml2/state-diagram.html