

Project Description

You are to develop a Finite State Machine for a simple elevator that operates between the ground floor and the first floor. The elevator is equipped with various sensors, control elements, and safety systems.

System Specifications

Elevator Cabin

- The elevator moves between two floors: Ground Floor (GF) and 1st Floor (1F)
- The cabin has automatic doors that open and close
- The elevator can transport a limited number of people

Available Sensors

- **Position sensors:** Detect whether the elevator is at the ground floor or 1st floor
- **Door sensors:** Monitor the state of the doors (fully opened/closed)
- **Weight sensor:** Measures the cabin load and detects overload
- **Timer systems:** Various time monitoring for safety and comfort

Control Elements

- **Call buttons:** Separate buttons for ground floor and 1st floor
- **Emergency stop button:** Immediate stop of the elevator in emergency situations
- **Reset button:** Reset after an emergency

Functional Requirements

Normal Operation

- The elevator starts in idle state
- Users can request the desired floor via call buttons
- The elevator travels to the requested floor
- The doors open automatically upon arrival
- After 5 seconds, the doors close automatically again
- The elevator returns to idle state

Safety Functions

- **Overload protection:** If weight is too high, doors remain open and elevator does not travel

- **Emergency stop:** Immediate stop when emergency stop button is pressed
- **Timeout monitoring:**
 - Doors must fully open/close within 10 seconds
 - Travel between floors may take maximum 2 minutes
- **Error handling:** In case of timeout errors, the elevator goes into emergency mode

Your Tasks

Task 1: State Identification

Identify all necessary states for this elevator. Consider:

- Idle state
- Movement states
- Door states
- Safety states

Task 2: Define State Transitions

Determine for each state:

- What conditions must be met to enter this state?
- What conditions lead to leaving this state?
- Which sensors and inputs are relevant?

Task 3: Create State Diagram

Create a complete state diagram with:

- All identified states
- All transitions between states
- Transition conditions (sensors, timers, buttons)
- Clear indication of the start state

Task 4: Scenario Analysis

Track the state transitions for the following situations:

Scenario A - Normal Journey:

- Elevator is at ground floor, user wants to go to 1st floor
- Describe each state transition from beginning to end

Scenario B - Overload:

- Elevator is at 1st floor, doors are open, too many people get in
- What happens and how is the problem solved?

Scenario C - Emergency:

- Elevator is traveling upward, someone presses the emergency stop button
- How does the system react?

Task 5: Implementation in IEC 61131-3 Programming Languages

Implement your Finite State Machine in all four IEC 61131-3 programming languages:

A) Structured Text (ST) - Two Variants:

- **Variant 1 - SR-Flipflops:** Use SET/RESET logic for each state with SR-Flipflops
- **Variant 2 - CASE Statement:** Implement the FSM with CASE statements and state variables
- Compare both approaches regarding readability and maintainability

B) Function Block Diagram (FBD):

- Use SR-Flipflops (SR function blocks) for each state
- Connect Set and Reset inputs with corresponding transition conditions
- Structure the logic into clear function blocks

C) Continuous Function Chart (CFC):

- Implement each state with SR-Flipflops
- Use free positioning for clear presentation
- Show data flows between SR-Flipflops

D) Ladder Diagram (LD):

- Use SR-Flipflops (Set/Reset coils) for each state
- Implement Set and Reset conditions as contact logic
- Structure each state as separate network

Implementation Notes:

- **SR-Flipflops:** Each state is represented by an SR-Flipflop
- **Set conditions:** Define when a state is activated
- **Reset conditions:** Define when a state is left
- **CASE structure (ST only):** Use enumerations for state variables

- Use consistent variable names in all implementations

Task 6: System Optimization

Analyze your design and answer:

- What additional safety features could be useful?
- How could the system be extended for three floors?
- What improvements would increase user comfort?

Helpful Tips

- Start with the simplest scenario (normal journey) and then expand with safety functions
- Remember that safety has top priority
- Consider that the elevator must always switch to a safe state in critical situations
- Use meaningful names for your states and variables
- Test your design mentally with various scenarios

Submission

Create complete documentation with:

1. State diagram
2. Description of all states and transitions
3. **Implementation in all four IEC 61131-3 programming languages:**
 - **Structured Text (ST)** - both variants: SR-Flipflops and CASE statement
 - **Function Block Diagram (FBD)** - with SR-Flipflops
 - **Continuous Function Chart (CFC)** - with SR-Flipflops
 - **Ladder Diagram (LD)** - with SR-Flipflops
4. Analysis of the three scenarios
5. Your optimization suggestions
6. **Comparison of implementation approaches:**
 - SR-Flipflops vs. CASE statement in ST
 - Advantages and disadvantages of each programming language for FSM implementation
 - When which approach is best suited

