

Programming an elevator using a PLC

Blake Tolmie, Rylie O'Brien

Friday J

ENMT221

2024

University of Canterbury

Due: 31st May, 11:55PM

Abstract

Programmable Logic Controllers (PLCs) are crucial to modern control systems, commonly including industrial machines and elevators. The development and testing of a model elevator used ladder logic and structured text in the CX-Programmer environment, emphasizing safety, efficiency, and performance. Initially, the elevator underwent a calibration state to accurately determine, calculate, and store the exact floor counter positions as word bits. Post-calibration, the elevator utilizes the SCAN disk scheduling algorithm to manage floor requests by transitioning between chill, going up, and going down states. This approach ensured the elevator was always on the optimal travel path to fulfil all requests, in the shortest time. The elevator's movement was controlled through proportional, integral, and derivative (PID) control, along with the use of a buffer region. User interface elements, such as buttons and lights, indicated the elevator's direction and the status of each floor request. The system successfully met performance criteria, filling requests within 3 minutes and achieving a travel time of under 30 seconds between the highest and lowest floors while ensuring safety and reliability. Future enhancements could involve utilizing more advanced PLC instructions for closed loop control and expanding the use of structured text for faster development.

Contents

Introduction	1
Methodology.....	1
Door controls	1
Calibration.....	1
Movement.....	2
Buttons and Lights	3
Scheduler	4
Results.....	5
Discussion.....	6
Movement	6
Scheduler	6
Buttons and lights.....	7
Conclusion.....	7
References	8
Appendix	9

Introduction

PLCs are the brains behind elevator control systems, controlling the operation of various components such as motors, doors, brakes, and safety systems. PLCs execute control logic programmed by engineers to ensure safe and efficient operation.

The use of PLC's is the industry standard for the purposes of an elevator control system. Key reasons include and are not limited to ease of repair and modularity. [1] The elevator is required to be safe. Safety can be defined as having doors only open when the elevator is on a floor limit switch. The elevator motion must move at a reasonable speed, there must not be a travel time between the highest and lowest floor of 30 seconds or greater. The elevator must fulfill all requests possible requests in an optimal manner, within 3 minutes.

Methodology

The sprints from the Scrum collaboration framework were incorporated into the development process to rapidly iterate on the development of the ladder logic code. [2]

Door controls

A safety state for the system was created to allow for safe operation of the elevators door controls, ensuring the door does not open while moving. The safety state activates after being on a floor limit switch and being stationary for 1 second. The safety state deactivates under the following conditions: the doors are closed, and a new target is set by the elevator's scheduler.

The door opening controls will function exclusively in the safety state. When the door safety state first activates once arrives at a floor, the doors are set to automatically open. The doors have an automatic closing timer set which closes the door after 6 seconds once the door has opened.

Calibration

Calibration was the first task that occurred in the elevator, finding floor target values for the movement system to use. The elevator moved from floor 1 to floor 5, back down to floor 3 where the elevator exited calibration state after the doors opened. After exiting calibration state, the elevator was free to take requests, and moves to floor 1 (chill state) where the elevator stays until a request is received.

The elevator performed this sequence of movements to calibrate the exact counter positions of each floor. When the elevator went to floor 1 during calibration and hit the first-floor limit, the high-speed counter was zeroed using the INI instruction. The zeroing of the counter sets a reference for all other floor counter values. The elevator then moved from floor 1 to 5, and back down to 3, where the exact counter position of floors 2, 3, and 4 were calculated by the averaging function block. The averaging function block took the counter position when the floor limit was differentiated up and down, then averaged to find the mid-point. This midpoint was used as the exact position of the floor.

Floors 1 and 5 limit counter positions were also stored during this process; however, they were not averaged because the elevator would trip the lower limit floor 1 alarm, and upper limit floor 5 alarm

when doing so. Instead, floors 1 and 5 had to have a manual offset from the limit switch counter value which was determined by testing with trial and error. This offset was different for the elevators on the simulator and real life.

Movement

A PID control was used to calculate the error for the system's transient response. A PID control system was developed because a closed-loop control system was needed for the elevator to function as intended. The proportional term is calculated based on the difference in encoder value between the destination floor and the position of the elevator.

The purpose of the integral control was to build up enough to maintain a small speed while decelerating while still reaching the target. If a fault were to occur resulting in the elevator falling short of the target, it will compensate over time and adjust to the correct target floor encoder value. This error is based on the distance the elevator is from the previous floor it was on. This error prevents steady state error from occurring, therefore assisting the elevator in reaching the target destination.

It was decided a differential control was needed to increase damping and improve stability. The purpose of differential control is to use proportional and integral control to predict where the system wants to accelerate and decelerate. This was calculated by recording the value of the encoder every 500 milliseconds and taking the difference between the 2 values.

The PID error value generated from the proportional, integral, and differential was multiplied by constants used for tuning and summed together to generate the total error, an approximation has been generated to show the intent of this error calculation, see Figure 1. The total error is calculated every 200 milliseconds and then applied to the motor to set its speed when the system is in a state which allows for movement. The real total error was found by taking a video recording of error values when testing the elevator, later the data was recorded at appropriate time intervals from the video.

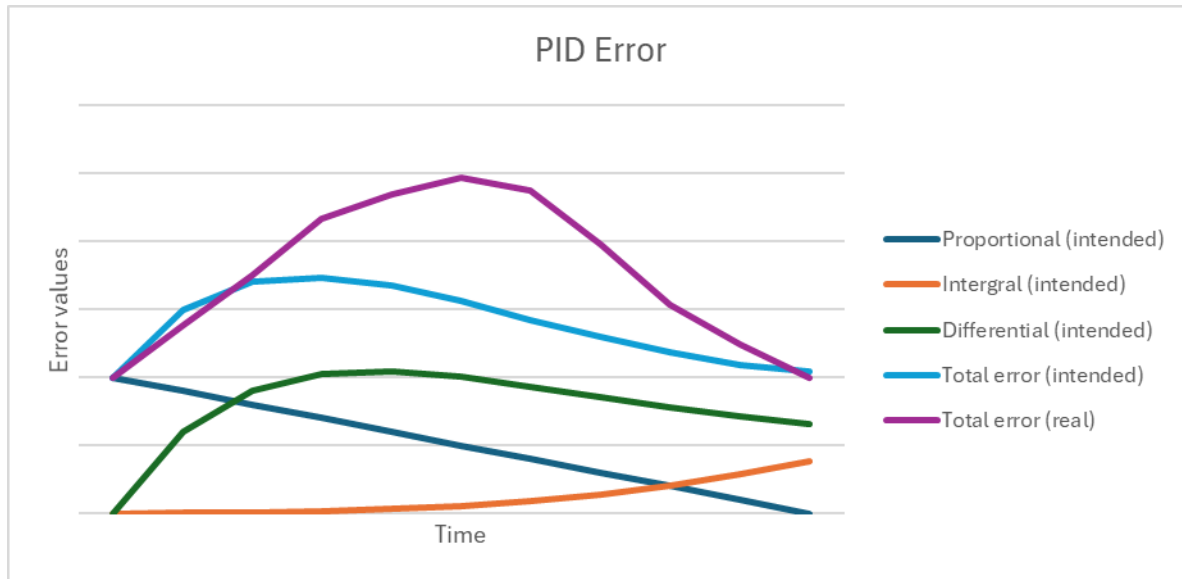


Figure 1 intended PID error calculation

In early development there were issues arising from the elevator carriage moving in unintended directions. To fix this the elevator carriage was made to always move in the direction of target. To achieve this, values for error were calculated based on absolute values. The error is then applied to the direction the elevator needs to move in to arrive at its target.

The motors applied 2 different speeds going up and down when the hold value was offset by the same amount. To address this inconsistency, it was decided two separate PID tunings were needed for the up and down states. As a result, the two separate PID calculations resulted in approximately the same speed whether going up or down.

A buffer region was created to have a maximum output on error to limit overshooting of the total error. The need for this arises from the error spikes from the real elevator that did not occur in the simulation from testing. The buffer region was further tuned to improve the PID motion to have a more desirable outcome.

Buttons and Lights

Once any main button, or floor 1 up, or floor 5 down on the elevator was pressed, the button light would turn on and stay on. The button would only turn off when the elevator reached the intended floor, and the doors opened. This was the same for floors 2,3, and 4, which had both up and down buttons. However, the button and light could only turn off if the elevator was moving in the same direction. Both up and down buttons on a single floor could only be turned off at the same time if there were no other floor buttons pressed. This was done to ensure that the scheduler algorithm would work, because the buttons acted like a floor request, only turning off when the request had been fulfilled.

External green indicator up and down lamps indicate the direction the elevator was moving in, when not on the desired floor.

Scheduler

The scheduler algorithm was based on the SCAN disk scheduling algorithm. The scheduler algorithm activated after calibration, and was always on. 'The SCAN disk scheduling algorithm operates by having the disk's head start at one end and move towards the other, servicing requests sequentially along the way. Upon reaching the end, the head reverses direction and continues to process requests as it moves back' [3]. Alongside SCAN algorithm's ensured balance and fair distribution of service across all disk requests, its ability to consistently service requests with minimal latency made it the most optimal algorithm for the elevator. The SCAN algorithm was implemented using a finite state machine outlined in Figure 2.

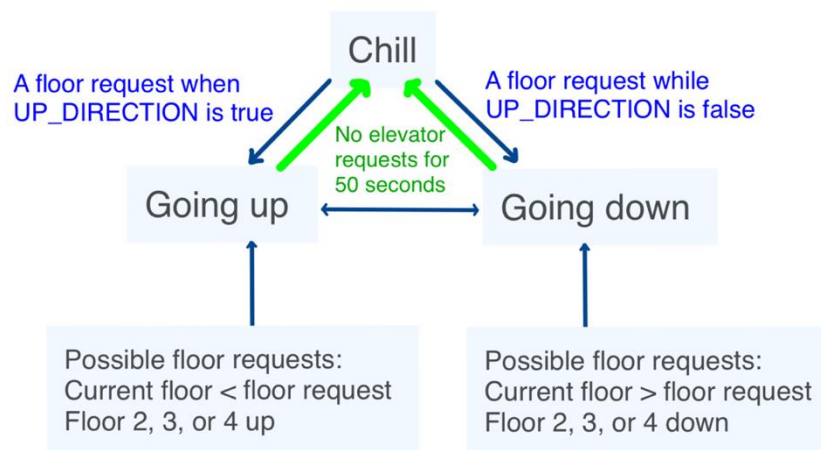


Figure 2 1Scheduler's finite state machine

A structured text function block 'SCHEDULAR' was used for defining the going up and down states, while ladder logic was used to define chill state. The implemented SCAN algorithm had the disk's head as the current floor, which was the last floor the elevator opened its doors on. In a typical SCAN algorithm, an array with all requests is separated into 2 lists 'left' and 'right' of the head, this scheduler used two ordered arrays, ascending going_up_list and descending going_down_list within the going up and going down states, as shown in Table 1. Where if a floor request was true, the floor number was assigned to a specific location in the array, for example a floor 3 request was indexed at 2 for both state arrays. If a floor was not requested, then 0 was assigned to it in the array.

Table 1 Table of the up and down state arrays when all buttons are pressed, and elevator is at floor 1.

State	Array	Floor request				
Going up	going_up_list	0	2	3	4	5
Going down	going_down_list	5	4	3	2	0
	Index	0	1	2	3	4

By having the arrays sorted in this way, depending on the state the elevator was in, the algorithm determined which floor to go to by choosing the closest floor request to the current floor. This diminished the time taken to fulfill all requests, always. The output from 'SCHEDULAR' function was then used to move the elevator to the desired floor.

The state of the elevator was controlled by when a floor request was received. When the elevator had any floor requests, the elevator could move into either a going up or going down state. The going up or going down state was defined by the last direction the elevator was moving in since the doors last opened, after leaving chill state. As in Table 2, this state can change depending on certain conditions.

Table 2 Definitions on when the going up or going down states change once requests have been made.

Going up → Going down	Going down → Going up	No change in state
If there are no floor requests going up in the going_up_list	If there are no floor requests in the going_down_list	Stays in going up state if there was a floor request going up below current floor in the going_up_list, but there are no floor requests in the going_down_list (current floor set to 0)
If there are no floor requests above current floor in the going_up_list	If there are no floor requests below current floor in the going_down_list	Stays in going down state if there was a floor request going down above current floor in the going_down_list, but there are no floor requests in the going_up_list (current floor set to 6)

In the going up state, the going_up_list is cycled through, and the closest floor request above the current floor is prioritized. Contrastingly, in the going down state, the going_down_list is cycled through, and the closest floor request below the current floor is prioritized. This accounted for a multitude of situations such as when the elevator was moving to a target floor, and the user wanted to stop at a floor on the way, then the elevator would stop at the floor on the way before target floor.

Chill state was defined to set the move target to floor 1, 50 seconds after there are no floor requests. The elevator then stayed in this position until a request was received.

Results

The motion control achieved the goal of having a travel time of no more than 30 seconds from the highest floor to the lowest floor, see Table 3. To measure the elevator was timed with a stopwatch from departure to arrival desired floor. The reason for measuring from the points in motion as opposed to doors opening is the subject of interest purely the motion, adding delays due additional layers in safety due to door control. During the same testing, there were no wait times greater than 3 minutes.

Table 3: time taken for elevator carriage to move between floors.

Moving between floors	Average time taken to reach floor while moving up (s)	Average time taken to reach floor moving while down (s)
1 and 2	8.75	9.59
1 and 3	14.16	15.74
1 and 4	16.74	19.98
1 and 5	21.14	26.45

Discussion

Movement

Due to the speed constraints using a buffer region the real and intended PID have different error outcomes, the real being a much more desirable pattern for a realistic elevator scenario. This justifies using a buffer region as it improves the motion pattern of the system.

Motion in the elevator going up and down have differences between them, despite having two separate PID tuning constant. The cause of this is the time-limited nature of the project, an improvement to the system would be having the elevator tune itself. There were restrictions in place for the purpose of this assignment not allowing the use of CX-Programmer's built-in instructions for implementing closed loop controllers. The relevant instruction for the project is an automated self-tuning PID controller as a single instruction block. This would be a more efficient solution for development time spent elsewhere compared to developing the PID and the time-consuming iterative process of tuning the PID.

Another limitation was restricting the use of structured text for only scheduling and finite state machine. Aspects of the system were much slower to develop in ladder logic as opposed to if it were to be developed in structured text, resulting in an opportunity cost of development time of other improvements for the system.

Scheduler

A previous limitation of scheduler was that the elevator could not take on requests once the elevator started moving. For example, if the elevator is moving up from floor 1 to floor 5, and a user presses floor 4, the elevator would keep going to floor 5. This was fixed by always having the scheduler function active, so the elevator could receive requests and change target destination while on the move.

The main limitation of the scheduler SCAN algorithm was request starvation, "the situation whereby a process must wait beyond a reasonable period of time—perhaps indefinitely—before receiving a requested resource" [4]. Request starvation was defined to be a user waiting longer than 3 minutes for their request to be served. The midrange floors 2-4 were typically serviced more than floors 1 and 5, meaning that floors 1 and 5 had the greatest potential to get request starved. To combat this, floor 1 was chosen to be the chill state for two reasons. The first being that in most buildings the greatest number of people enter the first floor, meaning the first floor will be the busiest and therefore have the

greatest number of requests. It was therefore essential to ensure that floor 1 didn't encounter request starvation, in any scenario. By having the elevator chill at floor 1 compared to other floors, it decreased the chance of request starvation for floor 1 and 5 requests, since the elevator would start at 1 then make its way up to floor 5 in the upstate. The greatest possibility for request starvation, occurred after the elevator moved from floor 1, with all buttons pressed. The new floor 1 request took under 3 minutes to be serviced during testing from Table 3, therefore starvation was determined not to be an issue.

Buttons and lights

One of the main problems encountered with buttons and lights was the directional buttons turning off when the elevator was moving in the opposite direction, causing request starvation. This is because the directional buttons acted like floor requests. By turning the button off before the floor request was serviced. For example, turning off the going up button at a floor when the elevator was going down the system falsely satisfied the request, causing request starvation as the elevator will never truly service the floor request. In contrast, if a button was not turned off when it was supposed to. For example, the going up light staying on at a floor after the request has been fulfilled then the elevator will keep fulfilling that request, resulting in request starvation for other floors.

One limitation was that floor 1 and floor 5 did not have 2 green indicator lights. This meant that floor 5 only indicated to the use when the elevator was moving down, and floor 1 when the elevator was moving up. This could be fixed by including 2 green indicator lights for floor 1 and 5. Or, the green indicator lights could indicate the elevator's next intended direction when at a floor.

Conclusion

A successful development and implementation of a PLC-based elevator control system was built using ladder logic and structured text in CX programmer. Leveraging the SCAN disk scheduling algorithm for efficient request management. The system achieved its primary goals of ensuring safety, with doors operating only when the elevator is in a safe state, filling requests within 3 minutes and maintaining a maximum travel time of under 30 seconds between the top and bottom floors.

Several enhancements were implemented to address initial limitations, such as enabling the elevator to take requests while in motion and improving the PID control to ensure acceptable speed in both directions. Despite encountering challenges like the potential for request starvation, particularly on floors 1 and 5, strategic decisions such as designating floor 1 as the chill state mitigated these issues.

Overall, the project demonstrates the effectiveness of using PLCs for elevator control systems, highlighting the advantages of modularity, ease of repair, and safety. Future improvements could include utilizing more advanced PLC instructions for closed-loop control and expanding the use of structured text for faster development. The system's robust performance underlines the viability of PLC-based control in modern elevator systems.

References

- [1] A. Becker, "Microcontroller based elevator controlling system," 2007 30th International Spring Seminar on Electronics Technology (ISSE), Cluj-Napoca, Romania, 2007, pp. 451-455, doi: 10.1109/ISSE.2007.4432898.

<https://ieeexplore.ieee.org/document/4432898>
- [2] Scrum.org. "What Is Scrum?" Scrum.org, 2020, www.scrum.org/resources/what-scrum-module.
- [3] SCAN (Elevator) Disk Scheduling Algorithms. (2019, July 29). GeeksforGeeks.

<https://www.geeksforgeeks.org/scan-elevator-disk-scheduling-algorithms/>
- [4] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts. Wiley.

Appendix

Trials for testing elevator speed:

	1 to 5	5 to 1	1 to 4	4 to 1	1 to 3	3 to 1	1 to 2	2 to 1
Trial 1	21.23	26.53	16.77	20.37	13.58	15.31	8.78	9.48
Trial 2	22.35	26.48	16.54	20.02	14.13	16.26	8.34	9.82
Trial 3	20.33	27.21	16.58	19.73	14.74	16.13	8.95	9.34
Trial 4	21.14	25.89	17.04	19.65	13.87	15.55	9.12	9.44
Trial 5	20.65	26.12	16.75	20.14	14.5	15.43	8.56	9.89
Average	21.14	26.446	16.736	19.982	14.164	15.736	8.75	9.594