ENMT211 Elevator Project Introduction

Term 2 2022

Note: This document will be amended if necessary throughout the term

1 Introduction

This project aims to provide you with practical experience in the design, development and testing of a PLC-based control system for industrial automation. It builds upon the PLC programming done in previous weeks.

Your task is to develop a PLC program to control an elevator model as summarized below. More detail will be provided in accompanying documents.

Summary of required behavior:

- Control the elevator motors so that the carriage moves smoothly but quickly between floors, stopping exactly at the desired floors
- Prevent users from opening the elevator doors while the carriage is between floors
- Control the call button light and floor indicator lights
- Implement a scheduling algorithm using a finite state machine:
    1. Determine the the sequence in which floors are to be visited
    2. Ensure no person using or waiting to use the elevator can be left waiting for ever
    3. Minimise waiting time

For full marks **realistic, complete, and reliable** elevator behavior is required.

2 Requirements

The project counts for 30 % total which comprises:

- 3 × 3 % **milestone assessments** in weeks 9 (from May 9), 10 (from May 16) and 11 (from May 23)
- A 12 % **group report** due Friday June 3, 11:59 pm
- A 7 % **final demonstration** in week 12 (from June 6)
- A 2 % **self reflection and peer assessment** due by end of week 12 (June 10, 11:59 pm)

Because of the short term, it is important to begin immediately and work steadily on the project throughout the term. If both team members attend and participate in the lab, they will usually receive the same mark for the milestones and final demonstration. The report is to be done as a group, where each team member is required to demonstrate understanding of and contribution to the system. **In the event of disputes/uneven contribution, I will determine marks for each team member**

3 Administration

3.1 Groups

You will undertake the project in pairs.
You must choose a partner from the same lab group. Anyone who has not chosen a partner by 11:59pm Wednesday May 4 will be auto allocated. In the event of an uneven number in a lab group, there will be a group of 3. **You must learn to work together, so I will not allow any 1-person groups**

The usual rules for collaboration when working in groups applies: feel free to help each other out, but don't do other groups' work for them.

3.2 Late demonstrations

Each milestones and the final demonstration must be carried out in the proper lab session (without exceptional circumstances e.g. medical note). If you expect to miss a lab for other reasons, we will attempt to work with you to arrange an earlier demonstration time. **No marks will be awarded for late milestones. Week 12 will be used for demonstrations only, with code submitted at the same time as the report.** Demonstrations during assessed labs will need to start at the beginning of the lab sessions - therefore you should already have completed the required functionality before the start of the lab where it is assessed. Late submissions on the final report will be penalised.

3.3 Hours of access

You may use the PLC lab so long as it is not in use for any other lab class and as permitted by the technician in charge. The elevator hardware will only be available for use during the scheduled lab B and C times, under TA supervision; at other times, the hardware will be locked. Failure to abide by these rules may result in penalties. Be aware that this assignment involves a considerable amount of work. It is unlikely that you will be able to complete the assignment using only the scheduled lab sessions. You will need to utilize your own time management skills to find time to work on the assignment outside these lab sessions. At all times, standard procedure for the lab applies: no food is permitted in the lab, only sipper-style water bottles no open-toed footwear, and **respect the physical distancing requirements of the current alert level**. Follow the instructions of the technician in charge (Dan Hopkins) at all times.

Depending on demand, there may be additional lab sessions where the elevators will be available and/or there will be access to elevator rigs in the final 2-3 weeks during working hours.

4 Hardware

The hardware for the project is permanently installed in the Controls Lab; the 'elevator' rigs roughly imitate the design of the elevators in the Civil/Mechanical building. Each rig is permanently wired to its accompanying PLC. You will not need to make any changes to the wiring or hardware for the project. The IO connections between the PLC and the elevator rig are described in Appendix A, and are also made available as PLC Symbols in the template file Template.cxp, which is available on Learn. If you find that one of these documents contains an error (i.e. doesn't correspond to how the elevator is physically wired up to the PLC), then bring it to the attention of one of the TAs so the documentation can be updated.

4.1 Allocation of Hardware

There are four elevator rigs. Nominally, all the rigs are identical, however there are some small variations due to production and component tolerances (see the reference to 'hold values' below). You will be able to select which of the elevator rigs you wish to use, so that you will only need to optimize your software to work on one particular rig. You are not required to only use one of the rigs; if a different rig is unused, you are free to it for testing. There are a maximum of 15 groups in each lab session. Therefore, of each 2 hour lab, you can expect to spend 20 minutes on the elevator. As in real life, it is important to plan your testing carefully to minimise the time required on the physical system. Most of your planning should be done using the simulator.

4.2 MCU Supervisor

To prevent unintentional damage to the hardware, each elevator is actually controlled by a built in microcontroller (MCU) (located somewhere in the bottom of the elevator). The MCU monitors the state of the elevator, and if an unacceptable situation is detected, then the MCU takes back control of the elevator from the PLC. In this case, the warning light on the top of the elevator flashes and the elevator carriage returns to the nearest floor. After around ten seconds, the MCU returns control of the elevator to the PLC. The most common example of this will be if you run the carriage into the limit switches at the top or bottom of the elevator shaft. Generally, doing something which causes the MCU to intervene is bad, and results in marks lost during the assessments. The only exception is that initially when your software starts up, it is allowable if it runs into either the top or the bottom limit switch once whilst it is calibrating itself.

4.3 Doors and Carriage

In order to move either the doors or the carriage, the enable line must be asserted. The enable line is wired so that it is asserted whilst a 'logic high' is output on the corresponding PLC output coil. If the enable line is not asserted, then commands to move either the doors or the carriage will be ignored. If the enable line is dropped whilst either the carriage or the doors are moving, then the MCU will take control of the elevator.

4.3.1 Doors

The motion of the doors is fully automated; your software just needs to send commands for 'open' and 'close'. The commands are issued by energizing the appropriate PLC digital outputs. Feedback is provided by limit switches, which indicate whether the doors are fully opened or fully closed. Once the door is fully open or closed, you must deactivate the command inputs (so that the motor is not left straining against the door stops) or the MCU will take control of the carriage. Attempting to both open and close the doors at the same time will also result in the MCU taking control of the carriage. To avoid jamming the lift, the MCU restricts when the doors are able to be opened: the lift must be on a floor (indicated by one of the floor limit switches being activated), and the carriage motor must be set to the appropriate 'hold value' (to ensure the lift isn't moving). Obviously, doors must be closed before moving away from a floor. The doors take some time to move, so you must send the 'close' command, then wait until the 'door closed' input signal is asserted.

4.3.2 Carriage

The motion of the carriage is controlled by the value of an analogue output; the motor controller is connected to DA channel 0 of the PLC. A voltage of 0V indicates full speed upward. A voltage of approximately 2.5V indicates zero speed (see below). A voltage of 5V indicates full speed downward. The exact voltage which results in the carriage not moving varies slightly from elevator to elevator (due to production and component tolerances). You will need to consider this if you run your software on different elevators. The 'hold value' is the hexidecimal value which you will need to send to the DAC in order to have the elevator stop moving. In order to satisfy the MCU, you need to use a value close to the hold value (where 'close' is defined by some mystery tolerance inside the MCU), to have control over the carriage doors enabled. The current hold values for each of the elevator rigs will be provided in the lab. Accordingly, you should set up your code so that when the elevator is stopped at a floor, you specifically set the DAC output to your chosen 'hold value'. If you find that when you do this, the elevator creeps really slowly in one direction, then you might have to tweak the hold value a little to fix this.

Elevator	Hold Value
1	0x0B6B
2	0x0B20
3	0x0B5A
4	0x0C41

IMPORTANT: The motor controller expects a value in the range 0-5V. You must set the output voltage range in the PLC settings to the 0-5V range to ensure that you do not output a voltage outside the expected range. If you fail to do this, and output too great a voltage, then you will blow the protection fuse in the rig (which will then be down for repair).

4.3.3 Position Encoders and Limit Switches

The elevator has two methods of determining where the carriage is located. Limit switches located at each floor indicate when the carriage is sufficiently close to a floor for the doors to work, and are wired to individual digital inputs to the PLC. A quadrature optical encoder is attached to the main drive motor, and provides an accurate position of the carriage. The encoder is wired into input pins on the PLC which are able to be configured as a high speed counter (high speed counter channel 0), to allow automatic quadrature decoding.

4.4 Buttons and Lights

The elevator control panel features a number of buttons and indicator lamps. There is a set of buttons which would normally be mounted inside the elevator carriage on a pendant wired to the front of the rig, and there are a number of buttons and indicators at each floor of the front panel of the elevator. Each button is connected to a separate digital input line on the PLC. The push buttons are wired such that they provide a 'logic high' signal whilst depressed. Each indicator lamp is connected to a separate digital output line on the PLC. The lamps are wired such that they are illuminated whilst a 'logic high' signal is output. Note that each push button also has an indicator lamp that can be activated by a digital output from the PLC. The buttons and indicators mounted on the PLC stand (which you utilized during the ENMT211 lab course) are not considered part of the elevator, and so are free for you to use for calibration, debugging or diagnostic purposes.

5 Software

Your task over the course of this assignment is to develop software to operate the elevator rig in as realistic a manner as possible. This includes, but is not necessarily restricted to:

- Determination of the floor to be visited next
- Movement of the carriage between floors
- Operation of the carriage doors
- Operation of indicator lamps

5.1 Programming Paradigms

CX-Programmer (the OMROM PLC programming IDE) offers two different methods of writing software ('programming paradigms') for your PLC: Ladder Programming and Structured Text. Ladder Programming is what you used during the Term 1 Lab Course. Structured Text is a text based (IEC-61131-3 compliant) language similar to PASCAL; a brief tutorial will show you how to use it in your PLC software. If you wish, you are allowed to implement your scheduling/finite state machine using the Structured Text paradigm. However, the rest of your software (including the motion controller) must be created using Ladder Programming. Some other PLCs don't offer a text based method of writing software, so it is important to be able to use Ladder Programming when required. You may wish to prototype any aspects of your code in any programming language you wish (Python, Matlab, etc.).

5.2 Interface Behavior and Doors

The buttons, lights and doors of the elevator should behave in a manner as similar as possible to a regular elevator. Some examples: When you press a call button, the button should light up, then go out when the elevator reaches that floor. The elevator doors should open upon reaching a floor, then close again after some time. The doors should absolutely not open whilst the elevator is moving, even if you press the 'open door' button.

5.3 Motion Control

You will need to implement some form of controller to regulate the motion of the carriage; the ideal motion is smooth, whilst being as fast as possible, and without overshoot. There are a number of different methods to do this, each with their own advantages and disadvantages (proportional, PI, hybrid

etc.). You will need to select a method which offers appropriate performance, yet is not overly ambitious to programme and debug in the available timeframe. Whatever you decide must be justified in your report. CX-Programmer offers a number of built-in instructions for implementing closed loop controllers (specifically, the PLC offers a totally automated self-tuning PID controller as a single instruction block). However, for the purposes of this assignment, you are not allowed to use these functions (or other related high-level functions) in the final version of your code (you may use them for prototyping, comparison, or system identification if you wish). For all assessments, you need to create your controller from scratch and describe its function.

5.4 Scheduling

A real elevator has some kind of algorithm which it uses to determine what to do. You will need to choose a suitable algorithm and implement this is software, using a clear and carefully finite state machine for clarity. Things to think about include:

- What does the elevator do when there are no requests?
- Will the elevator ever leave anyone stuck either outside or inside?

It is suggested you adopt a simple method to perform queuing and prioritization of multiple simultaneous requests, so that it is easy to check your design and implementation work without bugs. The "elevator algorithm" is a classic solution.

5.5 Software Engineering Considerations

The scope and complexity of this task means that, unless you put some forethought into the manner you go about designing and implementing your software, it will most likely turn into a gigantic mess. Further, if your program isn't clear and easy to understand, those who are marking your report will have difficulty following it and may have difficulty awarding marks. In contrast, if your software is well designed, it will be more reliable, faster to develop, and easier to maintain. Things which help make a program clear and easily readable by others include:

- Use of function-blocks, sections, tasks etc.
- Use of symbols.
- Good, clear commenting.
- Logical design and flow between sections.
- Careful allocation of addresses.

Good marks will require some kind of software engineering consideration. A tutorial on LEARN will explain how you can use certain functions of the CX-Programmer environment to aid in your software engineering. In addition, you will need to put some thought into how you will approach aspects such as hardware management, memory overflow and source control.

6 Report

The single largest component of this assignment is a technical report detailing your solution, which should be prepared and submitted as a team. This report should outline the nature of the assignment scenario, your approach to the solution (including software engineering and project management considerations), the way in which your programme operates and the design decisions behind this behaviour. It should assess the performance of your software (which you should test before the final demonstration) and comment on any limitations and areas where it could be improved. A more substantial guide for this report is provided.