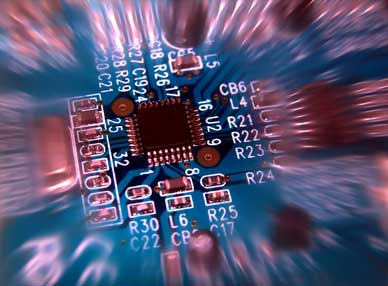
­

**ECSE 421**

**Group #5**

Max Chau   
Ching-Wai Chee   
Simon Foucher   
Jean-Mikael Lassonde   
Winston Lin   
Mathieu Perreault   
Logan Smyth   
Philip Tang   
Danny Wu



System Requirements Specifications Document

Table of Contents

[Section I: Scope 3](#_Toc253504645)

[Section II: Functional Requirements 4](#_Toc253504646)

[Main Requirements 4](#_Toc253504647)

[Error detection requirements: 6](#_Toc253504648)

[System Inputs and Outputs 8](#_Toc253504649)

[Section III: Technical Requirements 9](#_Toc253504650)

[High level design 9](#_Toc253504651)

[Hardware Requirements 10](#_Toc253504652)

[Software Requirements 10](#_Toc253504653)

# Section I: Scope

This project consists of an elevator system for a twenty floor apartment building with three elevators. The system will be made up of two distinct parts that will communicate over a serial line.

The first part is the monitoring system, and it will be run on an ordinary computer. It will handle the majority of the elevator logic and provide audio and visual feedback to simulate the three elevators. The system will also provide feedback for indicators such as the floor position signals and alarm signals.

The second part is the control system, and it will be run on a McGumps board consisting of a MSP430F149 MCU chip and a MAX7128AE PLD. This control system will be where the floor and elevator inputs are performed. The McGumps board will be connected to an input device and communicate the inputs to the monitoring system over a serial line. It will also retrieve information from the monitoring system as required.

Overall, the monitoring and response elevator system will behave as a simulator, as there are no real elevators involved.

# Section II: Functional Requirements

## Main Requirements

1. *There is one up/down signal per floor. Whenever one such button is pressed, the microcontroller responds by sending an elevator to that location with the intention of going in the direction signaled.*

With up and down buttons on each floor, when an elevator request button is pressed on the floor, the input signal will trigger an interrupt in the control system. The interrupt service routine will then parse the signal to determine which floor called for an elevator and its desired direction. This information will be sent to the monitoring system where it is processed through the dispatch algorithm to determine which elevator will serve the request. The closest elevator will generally serve the request in order to minimize wait time and maximize efficiency, unless it is already serving customers that are heading away from floor. The exact order of execution will be determined by the dispatching algorithm. Once the monitoring system decides which elevator will serve the request, it will send this information back to the control system to fulfill the task.

1. *Position feedback is sent by every elevator such that the control system is always aware of all the elevator positions.*

The control system will maintain the floor positions of every elevator in its memory. Whenever an elevator reaches a new floor, an event will be triggered in the monitoring system to send a message updating the status of the control system. This message will consist of the elevator that triggered the event and the new floor position. This event will be triggered even if the elevator does not stop on that floor and is only passing by. Once the control system receives the message, it will update its memory representation. Notice that this memory location keeping track of elevator position will be used for numerous logic comparisons by other tasks.

1. *Any given elevator spans all the floors of the building, such that any floor is accessible from all the others.*

The floors will be represented simply in the program logic. Whenever the program logic needs to change the floor values of a particular elevator, it will first verify if the value requested is within the range of the allowed integers. If so, the floor will be added to the elevator’s queue based on pre-defined priority, and the elevator will eventually get there. In order to make all the floors accessible from all the elevators, the same boundary restrictions will be used for all the elevators in the building. Maximum flexibility in the implementation will allow for changing the number of floors easily, if required.

1. *Each elevator is equipped with a number button board; one button representing one floor.*

When a user presses a button, the program logic receives a signal and sends that to the dispatching algorithm. Each elevator will have 20 buttons, one button for each floor. When a customer enters the elevator, he or she must push a button to send a request for the elevator to get to their desired floor. This button will trigger an interrupt on the control monitor. The interrupt service routine on the control monitor will then determine the button pushed and send this information to the monitoring system. Knowing that the button is pushed from inside the elevator, the elevator must travel to the desired floor independent of the other elevators’ status; therefore this request is added directly to the elevator’s queue. The monitor will send the signal back to the control system to serve the request. However, if the customer selects a floor that requires the elevator to travel in the opposite direction of its current task, the elevator must first finish the requests in the current direction, and then proceed to serve the request in the opposite direction.

1. *Each elevator is equipped with a EMERGENCY button.*

When a user presses the emergency button, the elevator goes and stops at the nearest floor and the doors are opened. This button will serve as a simple manual override within the elevator in case of emergencies. Furthermore, it will notify the control center that an emergency has occurred.

1. *Each elevator is equipped with an OPEN and CLOSE door buttons.*

Just like in most elevators, the primary motivation of these buttons is to enhance user control over the performance of the elevator. The ‘close doors’ buttons is only active when the doors are open. Instead of waiting for a timer before automatically closing its doors, the press of this button will bypass the timer and force the doors to close (as long as nothing is obstructing their way). The open button only works when the doors not shut. Its purpose is to either re-open the doors and reset the open door timer, or simply keep the doors open for an extended period of time. A typical use case for this button is if a user is inside the elevator and wish to keep the doors open for an extended period of time, or re-open them as they are closing to let a friend enter the elevator.

1. *A certain priority of commands will be maintained in the system.*

A certain priority of commands will be maintained in the system. Top priority will go to system emergencies handled by the system itself (for example if the speed sensor detects an abnormally fast elevator speed, it will issue a stop command which will interrupt the system and apply the emergency brakes on the elevator). The second priority will be given to the alarm button over the floor buttons. If some gets aggressed in the elevator and presses the alarm button while the ‘floor20’ was already selected, the elevator will still stop at the next floor and issue an alarm (instead of costing to the 20th floor, then react to the alarm button). Finally, the open door button will have priority over a floor select button. So for instance, if a user presses ‘floor x’ then immediately after (while the doors are still open) presses and holds ‘open doors’, the doors will remain open and the floor select will not be executed until the ‘open doors’ button gets released.

1. *A 3D visual interface will serve as visual support for the Elevator System.*

The system will contain a 3D interface to visualize the functioning of the Elevator System. The user will be able to see the numerous elevators currently in action. It will contain indicators to show what floors each elevator is on. The elevators will visually go up and down in the interface and a user will be able to see the doors opening and closing. The interface will distinctly display a jam of one of the elevators. It will also be able to view each one of the elevators' control panels separately. When a floor is called, the corresponding elevator's control panel will light up accordingly. The interface will be intuitive and easy to understand. No input controls (mouse or otherwise) will be supported on the interface, but the readings will come from the rest of the program logic to determine the position of the elevators.

## Error detection requirements:

1. *Very high speed recovery.*

If the speed received from an elevator’s velocity sensor exceeds the expected value by more than 10%, the emergency brakes will be applied to stop the decent. This will be simulated as a sudden increase in speeds reported by the sensor, which will trigger an immediate message to the monitoring system and apply the breaks. Whether it is caused by an overdriven/weak motor or inadequate breaking; it is a condition we want to avoid, and therefore will generate an alarm and ‘freeze’ the elevator by applying the emergency breaks. To get out of the situation, a manual reset for that elevator will have to be performed from the main control unit (the computer managing all of the elevators).

1. *Emergency Timeout*

If up/down button pressed without response within 5 minutes, the next elevator heading in the direction of that floor won't turn around until it reaches it. An alarm signal will also be activated (avoid starvation).This will be implemented by starting a counter whenever a given directional button on a given floor is pressed. When the counter reach a threshold (5 minutes), a starvation avoidance sub-routine will be called, otherwise if an elevator reach the floor and is going in the right direction, the counter is reset and wait to be activated again. The sub-routine will temporarily increase the priority of the said floor, so that the elevator will arrive faster. In case where multiple floors call this sub-routine, the priority will be set on a first come first serve basis. The sub-routine will return the floor to its usual priority level when an elevator reaches the floor in question.

## System Inputs and Outputs

**Input:**

* Desired floor position for each elevator (20 buttons per elevator)
* Direction of travel (up/down buttons on each floor)
* Emergency button (One button per elevator)
* Stop/Go button (One button per elevator)

**Outputs:**

* Desired floor signal (20 per elevator)
* Elevator direction (Up/down/stopped)
* Existing up/down requests on each floor

# Section III: Technical Requirements

## High level design



Figure - Prototype System Design

## Hardware Requirements

* McGumps Microprocessor Systems Board:
  + PS/2 Keyboard
  + Character LCD display
* Computer to act as a central processing station:
  + Keyboard
  + Monitor
  + Mouse
  + RS232

## Software Requirements

* Integrated Development Environment (IDE) :
  + CrossWorks
  + Quartus
  + Visual Studio
  + Notepad++
* Programming language :
  + C/C++ (Elevator microcontroller and Main control system)
* OpenGL for the Elevator Animation
* Communication protocols:
  + SPI
  + RS232