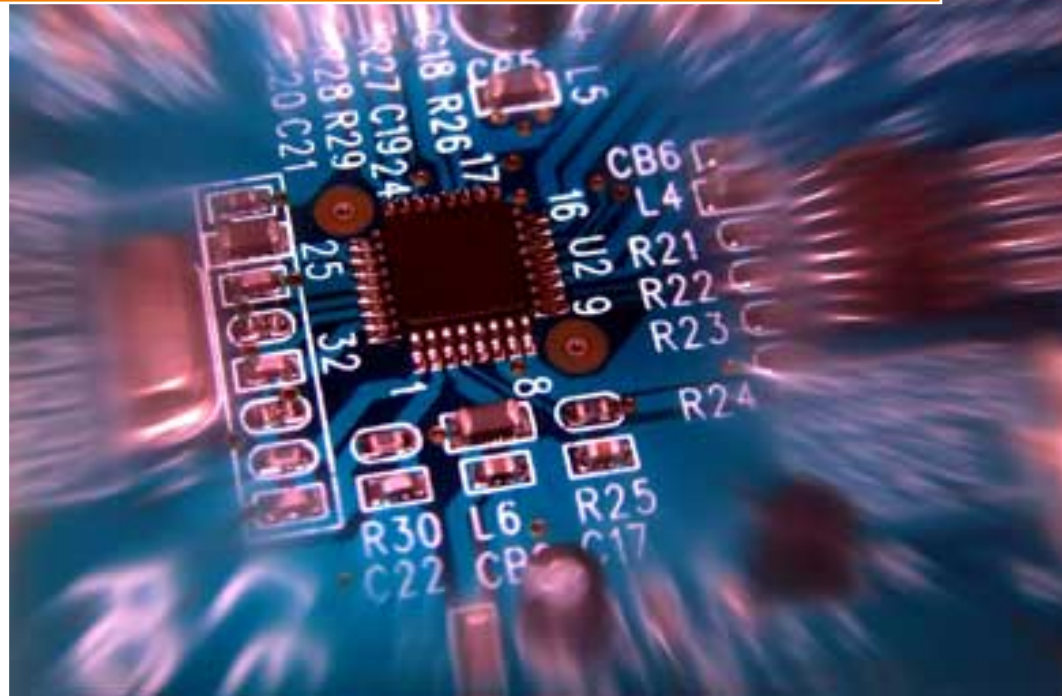


ECSE 421

System Testing Document



Group #5

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

Table of Contents

1. Introduction	2
2. Traceability Matrix.....	3
3. System Tests	4
Test 1: The Up/Down buttons	4
Test 2: Elevator position awareness.....	5
Test 3: Elevator service span.....	5
Test 4: Floor button board inside elevators	6
Test 5: The Emergency button	6
Test 6: Open/Close door buttons.....	7
Test 7: System command priority hierarchy implementation	8
Test 8: Visual support of virtual system	8
Test 9: Blackout recovery	9
Test 10: Emergency timeout.....	9
Appendix 1 - Elevator Instructions OpenGL.....	10
Appendix 2 - Elevator Instructions MSP430	11

1. Introduction

This document provides testing sequences for every functional requirement listed in the SDD, as well as an overview of the expected results of the elevator system project. This embedded system controls three virtual elevators servicing a twenty floor building and offers a friendly Graphical User Interface (GUI) to display the state of the elevators.

The system is divided into two major processing components which communicate via a serial peripheral interface (SPI) bus channeled over an RS-232 serial data cable. The first physical component, the central processing unit, is responsible for maintaining the state of the virtual elevators based on user Input, decision making, handles the SPI driver that receives commands from the second component, and controls the GUI which displays on a monitor the state of the system. It is implemented in hardware by a standard PC in a Linux environment, and the GUI is displayed on standard computer screen, using 'canned' openGL low-level drivers. The second major hardware component is implemented in a McGumps Microprocessor board (running an MSP430 micro-processor). Its responsibility is to capture all user input via a PS/2 keyboard and transmit them to the central processing unit. Since the McGumps board is simulating user input for the internal buttons of three elevators and the floor direction buttons on 20 floors, the system contains an LCD screen to display which system is currently being emulated.

2. Traceability Matrix

Requirements	SRS	SDD	STC	Does outcome match expected results?
1. The Up/Down Buttons	Sec 2.1	Sec 2.2.4, Sec 2.2.5	Sec 3.1	Yes
2. Elevator Position Awareness	Sec 2.2	Sec 2.2.2, Sec 2.2.3	Sec 3.2	Yes
3. Elevator Service Span	Sec 2.3	Sec 2.2.1, Sec 2.2.3, Sec 2.2.5	Sec 3.3	Yes
4. Floor button Board Inside Elevators	Sec 2.4	Sec 2.2.5	Sec 3.4	Yes
5. The Emergency Button	Sec 2.5	Sec 2.1, Sec.2.2.5	Sec 3.5	Yes
6. Open/Close Door Buttons	Sec 2.6	Sec 2.1, Sec.2.2.5	Sec 3.6	Yes
7. System command priority hierarchy implementation	Sec 2.7	Sec 2.2.2	Sec 3.7	Yes
8. Visual support of virtual system	Sec 2.8	Sec 2.2.3	Sec 3.8	Yes
9. Blackout recovery	Sec 2.9	Sec 2.2.2	Sec 3.9	Yes
10. Emergency Timeout	Sec 2.10	Sec 2.2.2	Sec 3.10	Yes

3. System Tests

The following section outlines a list of tests designed to challenge the implementation of all the proposed requirements. Each test outlines the requirements it is meant to test, the sequence of commands to send to the system as well as the expected results. To allow more flexibility while testing (in other words, to avoid proposing specific pre-selected test sequences and 'safe' actions), the test sequence has been written in high level manner. In order to convert the high level operations described into specific keystrokes understood by the system, please refer to Appendices 1 & 2 for a list of specific available keyboard commands.

Test 1: The Up/Down buttons

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

Implementation: Button low level drivers implemented in the McGumps Microprocessor board (See SDD Section 2.2.5). The event is handled in the central processing station via a serial I/O driver (See SDD Section 2.2.4)

Test sequence:

- 3.1.1 For floor 1, press the *up* button and wait for the elevator.
- 3.1.2 For floor 2, press the *up* button and wait for the elevator.
- 3.1.3 For floor 2, press the *down* button and wait for the elevator.
- 3.1.4 Do step 3.1.2 and 3.1.3 for floor number 3 to 19.
- 3.1.5 For floor 20, press the *down* button and wait for the elevator.

Expected Result:

For each *up* button pressed, an elevator will come to the floor that requested it with the intention to go up.

For each *down* button pressed, an elevator will come to the floor that requested it with the intention to go down.

Test 2: Elevator position awareness

Meets Functional requirement 2: *"Position feedback is sent by every elevator such that the control system is always aware of all the elevator positions."*

Implementation: Since the elevators are implemented virtually, its position will be generated by a sub system of the central processing station, responsible for creating, operating and maintaining the virtual elevators (See SDD Section 2.2.2). The elevator position feedback is also sent to the graphical user interface (GUI) which is displayed on the computer monitor (See section SDD 2.2.3). The communication is handled implicitly via a global variable elevator Object accessible by all systems.

Test sequence:

3.2.1 Send a request to the elevator to go to an arbitrary floor and observe its movement on the GUI.

Expected Result:

The elevator will be moving from its position to the requested floor in the GUI

Test 3: Elevator service span

Meets Functional requirement 3: *"Any given elevator spans all the floors of the building, such that any floor is accessible from all the others."*

Implementation: This is implemented at a low level in the Data Processing unit (See SDD Section 2.2.1) by means of variable boundaries. It is also reflected in the I/O drivers (See SDD Section 2.2.5): there are 20 valid floor buttons in each elevator which can be pressed by the user, as well as in the GUI (See SDD Section 2.2.3 the virtual building displayed is 20 floors high)

Test sequence:

3.3.1 Request the elevator to go to all floors from 1 to 20.

Expected Result:

Each elevator should stop and open at each floor

Test 4: Floor button board inside elevators

Meets Functional requirement 4: *"Each elevator is equipped with a number button board."*

Implementation: Implemented by the PS/2 keyboard connected to the McGumps Microprocessor board (See SDD Section 2.2.5)

Test sequence:

3.4.1 Request the elevator to go to an arbitrary floor using the keyboard

Expected Result:

The elevator should receive the request and go to the requested floor

Test 5: The Emergency button

Meets Functional requirement 5: *"Each elevator is equipped with an EMERGENCY button."*

Implemented in: Implemented by the PS/2 keyboard connected to the McGumps Microprocessor board (See SDD Section 2.1: System Architecture for Hardware and SDD Section 2.2.5 for software drivers)

Test sequence:

3.5.1 Press on the emergency button while the elevator is moving

Expected Result:

The elevator should stop at the closest floor, open its doors and will not accept request until it is given the instruction to unjam.

Test 6: Open/Close door buttons

Meets Functional requirement 6: *"Each elevator is equipped with an OPEN and CLOSE door buttons."*

Implementation: Implemented by the PS/2 keyboard connected to the McGumps Microprocessor board (See SDD Section 2.1: System Architecture for Hardware and SDD Section 2.2.5 for software drivers)

Test sequence:

- 3.6.1** Press on the open button while the elevator is moving.
- 3.6.2** Press on the close button while the elevator is moving.
- 3.6.3** Press on the open button while the elevator is stationary.
- 3.6.4** Press on the close button while the elevator is stationary.

Expected Result:

- 3.6.1** Nothing should happen; the elevator should not accept the requests. The elevator will stop at the next floor and open its doors if the open button is held.
- 3.6.2** Nothing should happen; the elevator should not accept the requests.
- 3.6.3** If the doors are closing or are closed, the doors should open.
- 3.6.4** If the doors are opening or are open, the doors should close.

Test 7: System command priority hierarchy implementation

Meets Functional requirement 7: *"A certain priority of commands will be maintained in the system."*

Implementation: Implemented in the Data Processing and decision making software component (See SDD Section 2.2.2)

Test sequence:

3.7.1 Simulate a power outage.

3.7.2 Emergency mode within elevator is active, press any command (except the button to disable the emergency mode).

3.7.3 Hold *open door* button.

3.7.4 Select a floor in the opposite direction of current travel.

Expected Result:

3.7.1 Elevator should go into emergency mode using backup power.

3.7.2 Elevator should not accept any request.

3.7.3 Elevator should not move while the doors are still open. The elevator will stop at the next floor and open its doors if the elevator is moving.

3.7.4. The elevator will finish servicing requests in the same direction and then proceed to service requests in the opposite direction of current travel.

Test 8: Visual support of virtual system

Meets Functional requirement 8: *"A 3D visual interface will serve as visual support for the Elevator System."*

Implementation: Implemented in the Graphical User Interface component (see SDD Section 2.2.3)

Same test and result as Test 2

Test 9: Blackout recovery

Meets Functional requirement 9: *“In case of a power outage, the system has a ‘recovery mode’ to set all elevators to ‘emergency mode’.”*

Implementation: Implemented in the Data Processing and decision making software component (See SDD Section 2.2.2)

Test sequence:

3.9.1 Same as 3.7.1

Expected Result:

3.9.1 Same as 3.7.1

Test 10: Emergency timeout

Meets Functional requirement 10: *“The system is equipped with an emergency timeout mechanism to avoid starvation of any requests.”*

Implementation: Implemented in the Data Processing and decision making software component (See SDD Section 2.2.2)

Test sequence:

3.10.1 Press up or down button and wait for more than five minutes.

Expected Result:

3.10.1 After five minutes, an alarm signal will activate.

Appendix 1 - Elevator Instructions OpenGL


There are two ways of controlling the elevators: directly from the GUI (Using the commands outlined below), or from the MSP430 keyboard (Using the commands outlined in Appendix 2)

Different sets of buttons have been assigned to different elevators and have been divided in columns (see table below). In order to make a floor selection, first scroll to it using the appropriate buttons, referring to the "Requested Floor:" label to know the current selection. When the desired floor appears besides the Request Floor button, press the floor select button to send out the request.

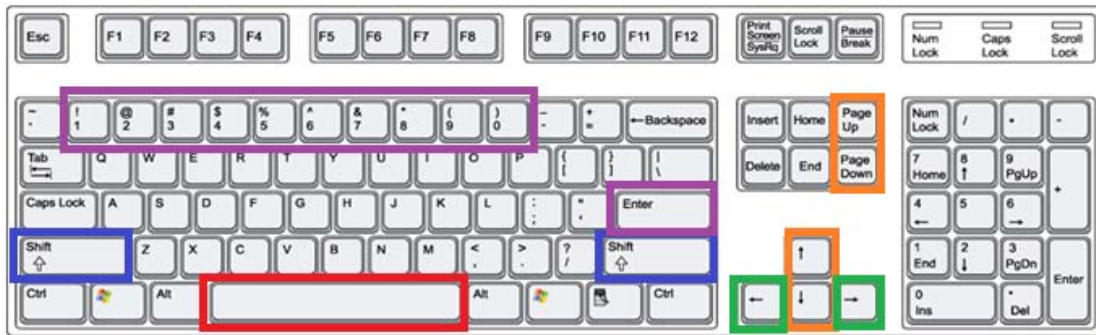
The specific function of each individual button is outlined in the following table:

	Elevator 1	Elevator 2	Elevator 3
Increment Floor Selection	Q	W	E
Decrement Floor Selection	A	S	D
Select current floor Selection	Z	C	C
Emergency Button	1	2	3
Open Doors	shift+Q	shift+W	shift+E
Close Doors	Shift+A	shift+S	shift+D

For the buttons in the hallway, first select the button panel floor by pressing 'R' to increment the floor selection, and 'F' to decrement the floor selection. The current selected floor is written besides the "UP/DOWN – Floor xx" label, where xx refers to the floor currently being operated on. Once on the desired floor, press V for UP and B for down (see table below).

Controls	Floors
	<p>First select the floor to operate on using: R,F: Increment/Decrement floor selection</p> <p>Then press: V: Press UP button on selected floor B: Press DOWN button on selected floor</p>

Appendix 2 - Elevator Instructions MSP430



Keyboard Keys	Actions
General options	
	Toggle between elevators
	Toggle between floors
	Floor, elevator request Up and Down
Elevator options for inside	
	Choosing floor number
	Enter Confirm floor number
	Left Open door
	Right Close door
	Space bar Emergency stop button