

## **SYSC 3303- Elevator Control System- Final Report**

**Prepared for:**

Shikharesh Majumdar

**Group Number:**

5

**Prepared by:**

Ryan Frohar -101029053

Xander May - 101038396

Andrew Cowan - 100882240

Ryan Gaudreault - 100968218

Kaelan Leroux - 101008680

**Link to GitHub Repository:** <https://github.com/godrowr/elevator-simulator>

## Table of Contents

<b>Responsibilities:</b>	<b>3</b>
<b>Set-up Instructions:</b>	<b>3</b>
<b>Table 2: File Description</b>	<b>4</b>
<b>Diagrams:</b>	<b>5</b>
State Machine Diagrams:	5
UML Class Diagrams:	6
Sequence Diagrams:	7
<b>Elevator Data:</b>	<b>8</b>
<b>Reflections:</b>	<b>9</b>

## Responsibilities:

Group Member	Role
Ryan Frohar	Ryan was responsible for writing code related to the Elevator, implementing UDP, bug fixing and participated in many group coding sessions
Xander May	Xander was responsible for creating the state machine in the Scheduler, updating much of the code between iterations and participating in many group coding sessions
Andrew Cowan	Andrew was responsible for writing code related to the Scheduler, implementation of UDP, collecting data for the elevator requirements, and participating in group coding sessions
Ryan Gaudreault	Ryan was responsible for the development of the Floor Subsystem, implementation of distributed functionality, testing and participated in many collaborative coding sessions
Kaelan Leroux	Kaelan was responsible for producing and updating the UML class and sequence diagrams. Kaelan was integral in the design of the code structure and participated in many coding sessions

**Table 1:** Team Member Responsibilities

Note: Much of the code was written collaboratively between multiple group members. The final report was prepared as a group using collaborative word editing and video conferencing.

## Set-up Instructions:

Import the eclipse project and run the Main file. The elevators should print their current position and whether their doors have been opened or closed. The elevators move independently throughout the floors to respond to the requests of the passengers. The path of the elevators is defined in the inputFile.txt.

Due to the lack of resources relating to the capture and recording of screen video, a video was not produced. Due to the lack of video, step-up instructions, and a detailed description was provided.

A brief explanation of each file in the project is provided in *Table 2*.

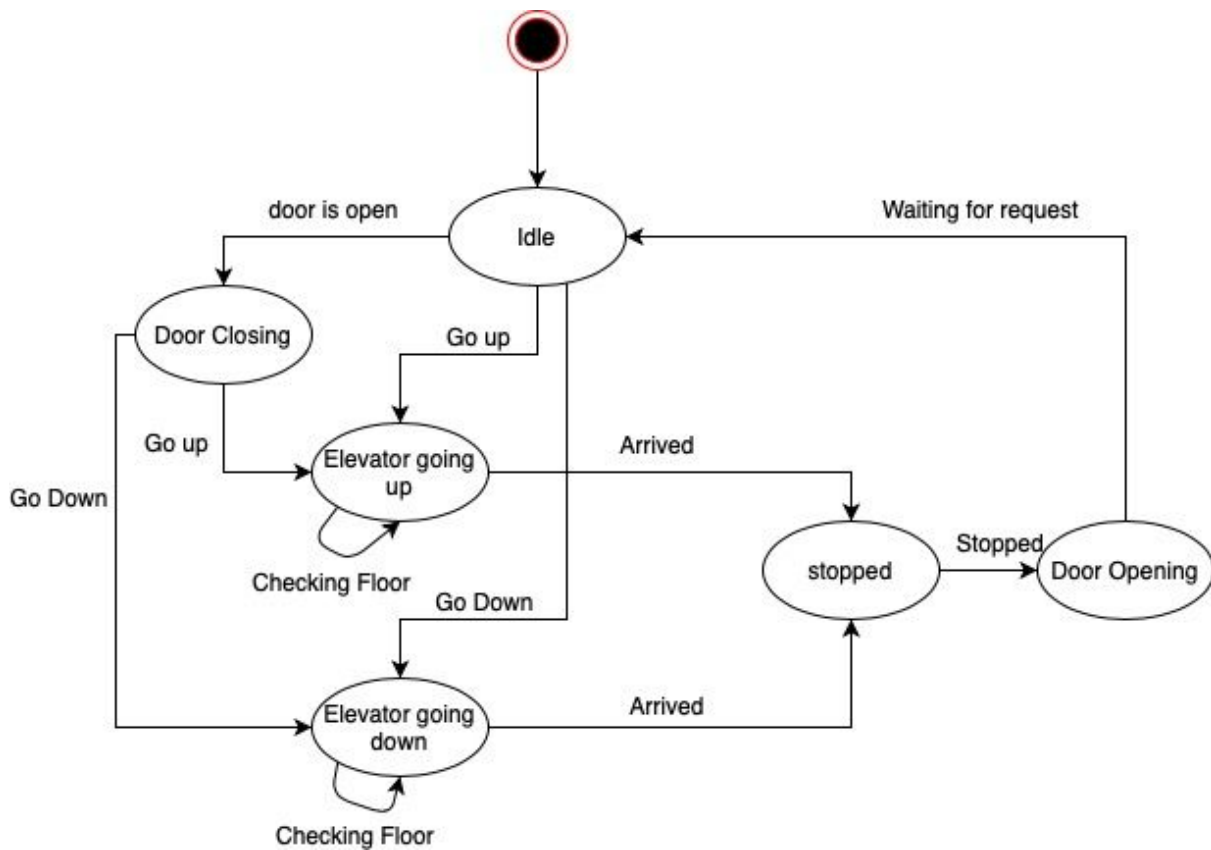
File Name	Description
FloorSubsystem	This file is the floor subsystem of the elevator simulation.
FloorTest	This file tests the floor methods.
ElevatorSubsystem	This file is the elevator subsystem of the elevator simulation.
ElevatorSubsystemTest	Thus file tests the ElevatorSubsystem and elevator methods.
Scheduler	This file is the scheduler subsystem of the elevator simulation.
SchedulerTest	This file tests schedule's methods.
Button	This file includes the button classes for the elevator and the floor.
ArrivalSensor	This file (not complete) includes the arrival sensor class.
Lamp	This file includes the lamp classes of the floor and elevator.
Main	This file when run starts the Elevator_subsystem, FloorSubsystem and Scheduler threads and showcases the information that is sent between the two.
inputFile	This file consists of the format which is indicated in the project requirements. It includes the date, floor number, direction and destination of each request.
Buffer	This file holds an array of RecvData objects to be easily iterated through and acquired.
RecvData	This file holds the data structure that is passed from one subsystem to another
UDP	This file creates sockets for the system that passes packet data to subsystems.

**Table 2:** File Description

## Diagrams:

Each iteration involved many diagrams for reader understanding. Unified Modelling Language (UML) diagrams are extremely important for communication in software design. Class diagrams, state machines and sequence diagrams of the project can be found in *Figure 1*, *Figure 2* and *Figure 3* respectively.

## State Machine Diagrams:



**Figure 1:** State Machine Diagram

```

classDiagram
    class Scheduler {
        <<implements Runnable>>
        Direction : enum
        - floorRequest: ArrayList<FloorButton>
        - floorSubsystem: FloorSubsystem
        - start: Instant
        - buffer: Buffer
        - uDP: UDP
        + Scheduler():
        + querySubsystem(): void
        + getNextFloor: ElevatorButton {Guarded}
        + run(): void
        + decodeMsg(RecvData): int[]
    }

    class Buffer {
        - storage: ArrayList<RecvData>
        - empty: boolean
        + Buffer():
        + add(RecvData): void{Guarded}
        + get(): RecvData{Guarded}
    }

    class Main {
        + main(String[]): static void
    }

    class ElevatorSubsystem {
        - elevators: ArrayList<Elevator>
        - threads: ArrayList<Thread>
        + Elevator_subsystem(Scheduler, int):
        State: enum
    }

    class Elevator {
        <<implements Runnable>>
        - buttonList: List<ElevatorButton>
        - currFloor: int
        - motor: Motor
        - door: Door
        - scheduler: Scheduler
        - state: State
        - elevatorNo: int
        + Elevator(Scheduler, int):
        - nextStop(): int
        + run(): void
        + update(): void
        + gotoFloor(int): void
        - decodeMsg(byte[]): ElevatorButto
    }

    class ElevatorInfo {
        - dest: int
        - elevatorNo: int
        - current: int
        + ElevatorInfo(intent, int):
        + getElements(): int[]
    }

    class RecvData {
        + port: int
        + data: byte[]
    }

    class ElevatorButton {
        + ElevatorButton(int, int)
        + getFloor(): int
        + getDest(): int
    }

    class FloorButton {
        - time: Instant
        + FloorButton(String, int, String, int):
        + getTime(): Instant
        + getFloor(): int
        + getDes(): int
    }

    class UDP {
        - receivePortNum: int
        - sendPortNum: int
        - iPAddress: InetAddress
        - sendReceiveSocket: DatagramSocket
        - receivePacket: DatagramPacket
        - sendPacket: DatagramPacket
        + UDP(intent, InetAddress):
        + initializeSocket(): void
        + receive(): RecvData
        + sendByte(byte[]): void
    }

    class FloorSubsystem {
        FloorType: enum
        - NUMFLOORS: static final int
        - floors: List<Floor>
        - buttons: ArrayList<FloorButton>
        + Floor_subsystem():
        + parseFile(): void
        + getRequest(Instant): ArrayList<FloorButton>
        + getFloors(): list<Floor>
    }

    class Floor {
        - floorNo: int
        - buttonList: List<FloorButton>
        + Floor(int, FloorType):
        + getFloorNo(): int
        + getType: FloorType
    }

    class Door {
        - open: boolean
        + Door():
        + toggle(): void
        + isOpen(): boolean
    }

    class Motor {
        - DELAY: final long = 2000
        - travelling: boolean
        + Motor():
        + travelNum(int): void
        + istravelling(): boolean
    }

    class Lamp {
        - on: boolean
        + toggle(): void
        + isOn(): boolean
    }

    class FloorLamp {
        + FloorLamp():
    }

    class ElevatorLamp {
        + ElevatorLamp():
    }

    class ArrivalSensor {
        + ArrivalSensor():
        + getCurFloor(): int
    }

    class Button {
        - push: boolean
        # lamp: Lamp
        # floorNo: int
        # dest: int
        + Button(int, int):
        + pushButton(): void
    }

    Scheduler --> Buffer
    Scheduler --> Main
    Scheduler --> ElevatorSubsystem
    Scheduler --> Elevator
    Scheduler --> ElevatorInfo
    Scheduler --> RecvData
    Scheduler --> ElevatorButton
    Scheduler --> FloorButton
    Scheduler --> UDP

    Buffer --> Scheduler
    Main --> Scheduler
    ElevatorSubsystem --> Scheduler
    ElevatorSubsystem --> Elevator
    ElevatorSubsystem --> ElevatorInfo
    ElevatorSubsystem --> RecvData
    ElevatorSubsystem --> ElevatorButton
    ElevatorSubsystem --> FloorButton
    ElevatorSubsystem --> UDP

    Elevator --> Scheduler
    Elevator --> Buffer
    Elevator --> ElevatorInfo
    Elevator --> RecvData
    Elevator --> ElevatorButton
    Elevator --> FloorButton
    Elevator --> UDP

    ElevatorInfo --> Scheduler
    ElevatorInfo --> Buffer
    ElevatorInfo --> Elevator
    ElevatorInfo --> RecvData
    ElevatorInfo --> ElevatorButton
    ElevatorInfo --> FloorButton
    ElevatorInfo --> UDP

    RecvData --> Scheduler
    RecvData --> Buffer
    RecvData --> Elevator
    RecvData --> ElevatorInfo
    RecvData --> ElevatorButton
    RecvData --> FloorButton
    RecvData --> UDP

    ElevatorButton --> Scheduler
    ElevatorButton --> Buffer
    ElevatorButton --> Elevator
    ElevatorButton --> ElevatorInfo
    ElevatorButton --> RecvData
    ElevatorButton --> FloorButton
    ElevatorButton --> UDP

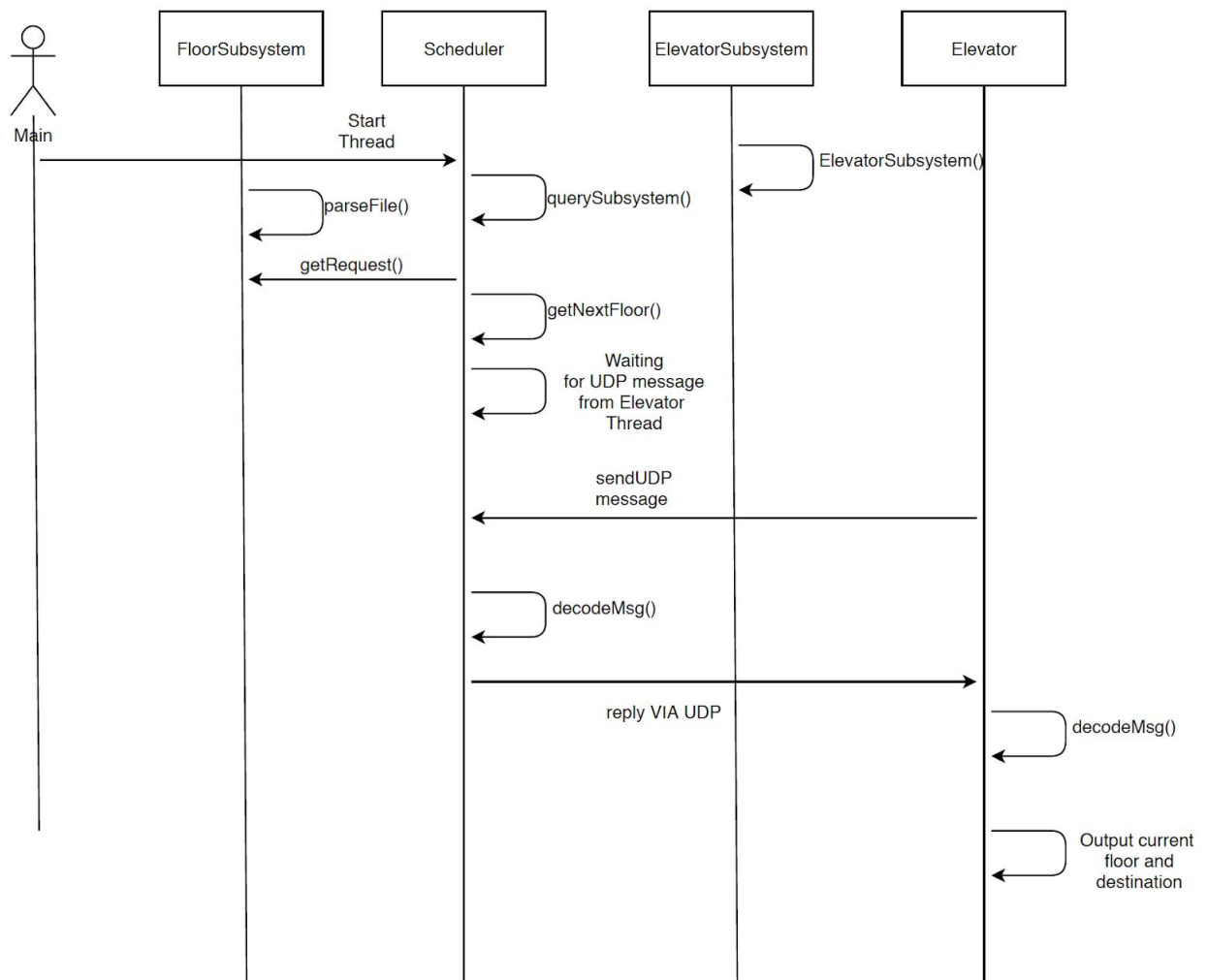
    FloorButton --> Scheduler
    FloorButton --> Buffer
    FloorButton --> Elevator
    FloorButton --> ElevatorInfo
    FloorButton --> RecvData
    FloorButton --> ElevatorButton
    FloorButton --> UDP

    UDP --> Scheduler
    UDP --> Buffer
    UDP --> Elevator
    UDP --> ElevatorInfo
    UDP --> RecvData
    UDP --> ElevatorButton
    UDP --> FloorButton

    Scheduler --> FloorSubsystem
    FloorSubsystem --> Scheduler
    FloorSubsystem --> Floor
    FloorSubsystem --> Door
    FloorSubsystem --> Motor
    FloorSubsystem --> Lamp
    FloorSubsystem --> FloorLamp
    FloorSubsystem --> ElevatorLamp
    FloorSubsystem --> ArrivalSensor
    FloorSubsystem --> Button
  
```

6

## Sequence Diagrams:



**Figure 3:** Sequence Diagram

## Elevator Data:

The data collected in iteration 0 is included in this report. The measurements can be found in *Table 2* and *Table 3*. In the environment that the measurements were gathered, there were 14 steps per floor and each floor was 98 inches in height.

Run	1	2	3	4	5	6	7	8	9	10	11	12
Floors												
1	0	0	0	0	0	0	0	0	0	0	0	0
2	3	3.5	3									
3				6	5.5	6						
4							8	8	8.5			
5										10	11	10.5

**Table 3:** Results From Elevator Measurements

Run #	1	2	3	4	5	6	7	8	9	10	11	12
Velocity (in/s)	32.7	28.0	32.7	32.7	35.6	32.7	36.8	36.8	34.6	39.2	35.6	37.3
Acceleration (in s <sup>2</sup> )	10.9	8.0	10.9	5.4	6.5	5.4	4.6	4.6	4.1	3.9	3.2	3.6

**Table 4:** Velocity and Acceleration Measurements



Min	5	Seconds
Max	6	Seconds
Average	5.4	Seconds
Std Dev	0.49	Seconds

**Figure 4:** Open/Close Measurements

Min	5.5	Seconds
Max	7	Seconds
Average	6.2	Seconds
Std Dev	0.51	Seconds

**Figure 5:** Additional Open Time Statistics

## Reflections:

In reflection, we are proud of many aspects of our work and are interested in developing further Java applications in the future.

When performing projects with long term goals in the future, it is critical that the final goals of the project are made clear at the beginning. Some of the code had to be re-written or significantly modified in order to enable further features. Skeleton code of the final project should be implemented in the first iteration to facilitate seamless addition of missing features.

One of the challenges with the project was coordinating six group members with varying school and work schedules. In future, an agile approach should be taken to ensure that all group members are up to date on the current progress on the project and have defined goals. Sprints should occur between the project deliverables and progress should be tracked using JIRA.

The group consisted of members with a diverse background of work, and educational experience. Many of the group members were able to contribute in their areas of expertise and elevate the knowledge of the other group members.