|  |
| --- |
| Rochester Institute Of Technology |
| Garage Door Opener |
| Design Document |
|  |
| **JENNY ZHEN & JUSTIN COTNER** |
| **3/18/2014** |

Table of Contents

[Introduction 2](#_Toc381714483)

[Class Diagrams 3](#_Toc381714484)

[Overall Class Diagram 3](#_Toc381714485)

[Sequence Diagrams 4](#_Toc381714486)

[Sequence Diagram - Door Open 4](#_Toc381714487)

[Statecharts 5](#_Toc381714488)

[Overall Statechart 5](#_Toc381714489)

[Overall Statechart Comments 6](#_Toc381714490)

[Input Statechart 7](#_Toc381714491)

[Input Statechart Comments 7](#_Toc381714492)

[Motor Statechart 7](#_Toc381714493)

[Motor Statechart Comments 7](#_Toc381714494)

[General Comments 8](#_Toc381714495)

# Introduction

The purpose of this design document is to describe the design of a controller embedded into a garage door opener. It is a fairly simple system that has two primary functions, open the door and close the door. The controller also has a number of safety features. When the door starts to close, an infrared sensor placed close to the ground is activated. If this beam is broken, the door stops and rises back up. The other safety concern that the controller watches for is a motor overcurrent. When an overcurrent is detected the motor stops and reverses direction if the door was closing and just stops if the door was rising.

# Class Diagrams

## Overall Class Diagram



In the system, the Input class takes user input and passes it to the Controller class. The Controller class scans the values of the control variables every second (1 Hz). If the control variables are modified, the Controller calls the appropriate methods in the Motor class.

Sequence Diagrams

## **Sequence Diagram - Door Open**



# Statecharts

## Overall Statechart



### Overall Statechart Comments

The overall system starts with the door in the closed state. If the button is pushed, it begins raising the door for up to 10 seconds. If the transition between opening and opened is interrupted, the motor turns off and goes to the paused opening state. Once the button is pushed again, the door can only close. If that transition, from closing to closed, is interrupted, the door goes back to opening.

When the door starts to close the infrared beam is turned on. If the beam is interrupted the door reverses direction and begins opening again.

In the worst case, both the up and down signals are given and the system goes into an error state. The conditions to get out of the error state were not specified in the requirements.

## Input Statechart



### Input Statechart Comments

The input processing for this system goes from waiting to parsing, and parsing to waiting. When there is input given, it modifies the control variables as necessary.

## Motor Statechart



### Motor Statechart Comments

The motor goes up and down based on input passed through the controller. If the motor receives input to turn the motor up and down simultaneously, it enters the error state.

The transition to get out of the error state for the motor was not specified in the requirements. As a result, the motor can never get out of the error state once it enters it.

# General Comments

Our first impression of using Rhapsody as a tool for our design needs is that Rhapsody provides the necessary features to create class diagrams, sequence diagrams, and statecharts.

Some of the things we disliked about Rhapsody include:

* The way Rhapsody itself was designed was not user intuitive, or user friendly. For example, to indicate multiplicity between two directed classes on a class diagram, one would need to choose a general “association” rather than a “directed association.” Then, putting in the multiplicity after the selection, and then double clicking into the dialog box, and finally unchecking “navigation” on the end of the association with the arrow.
* There were many bugs that we encountered. An example is for class diagrams, that one would need to right click to add an attribute or operation to a class. If one would double click into the dialog box, and try to add the attributes there, it will not appear in the diagram.
* The system we designed is small enough to be manageable using Rhapsody. If a bigger system had to be designed, it probably wouldn’t have worked as well.

Overall, Rhapsody worked well enough for the work we had to accomplish.