R·I·T

Rochester Institute of Technology

STATE MACHINE HARDWARE SIMULATION

Modeling of Real-Time Systems
(CMPE 564)

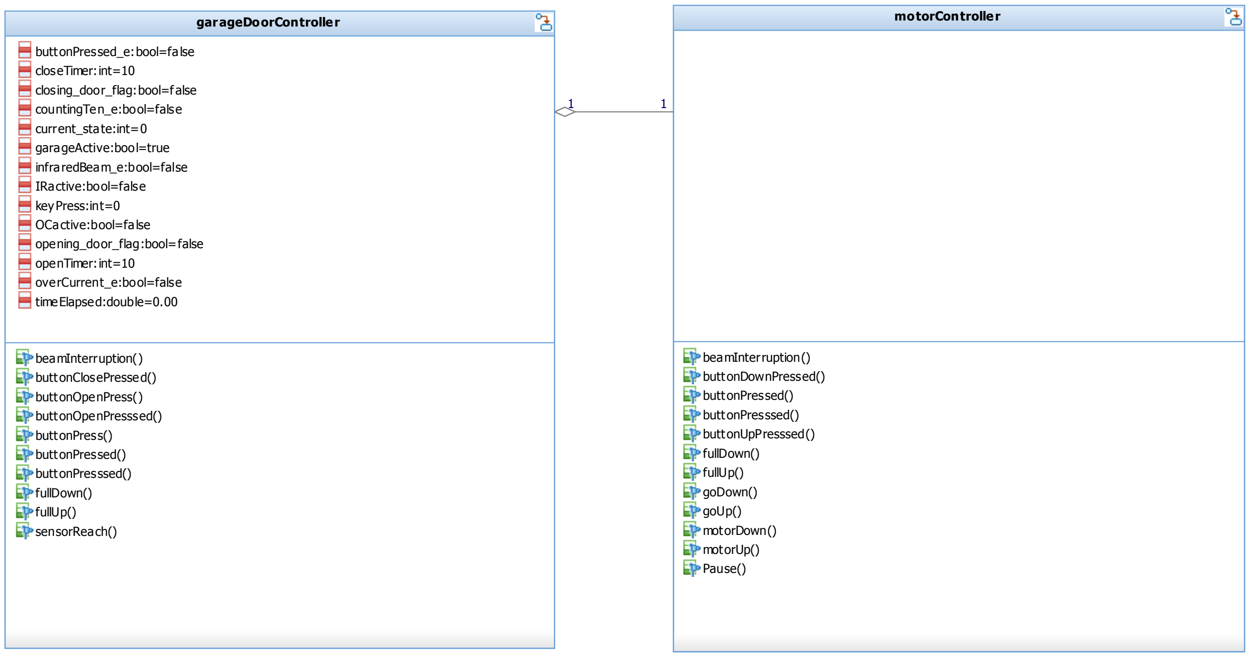
Alberto J. Santos-Castro
UID: 572009346

Thursday April 6, 2016

1. Introduction

In this stage of the project the main purpose was to include the hardware to the previous garage door controller state machine. The simulation of the sensors and actuators was done by the FPGA and supplied by the instructor. The program should be able to switch between hardware simulation and software, using the "s" command through the console. For the hardware simulation, the program should start with a reset to the FPGA to verify that both systems are on the same state. After this first stage the state machine should respond to the button commands in a timely manner. The FPGA would also report any problems with the implementation if it doesn't go to the adequate state or is not real-time, for instance, motor up and down at the same time. The software simulation should not be much different from last assignment, only the "s" command to switch simulation mode.

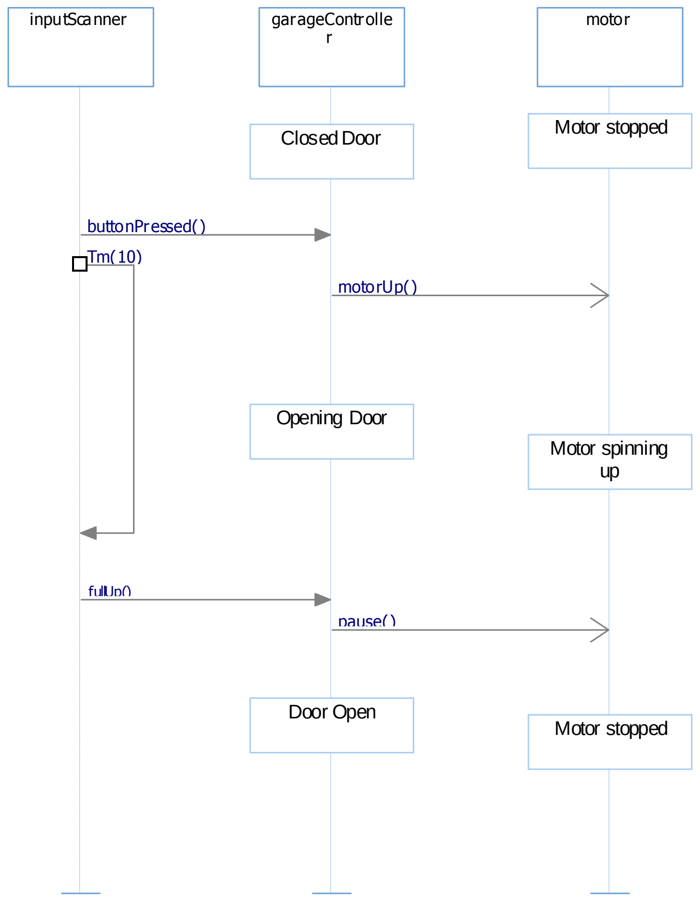
2. Class Diagram (discuss design highlights)



The two main classes from the first project were kept. Their relationship did not change, only the methods inside those classes changed. Also, a couple modifications were made to rectify the behavior with an overcurrent interruption.

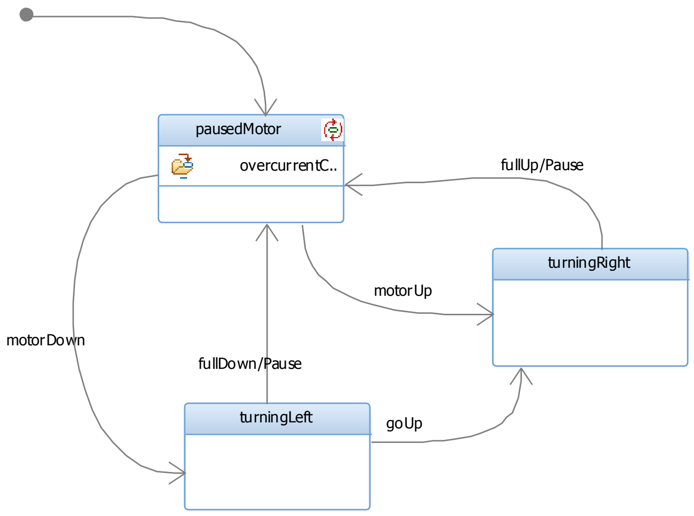
3. Sequence Diagram:

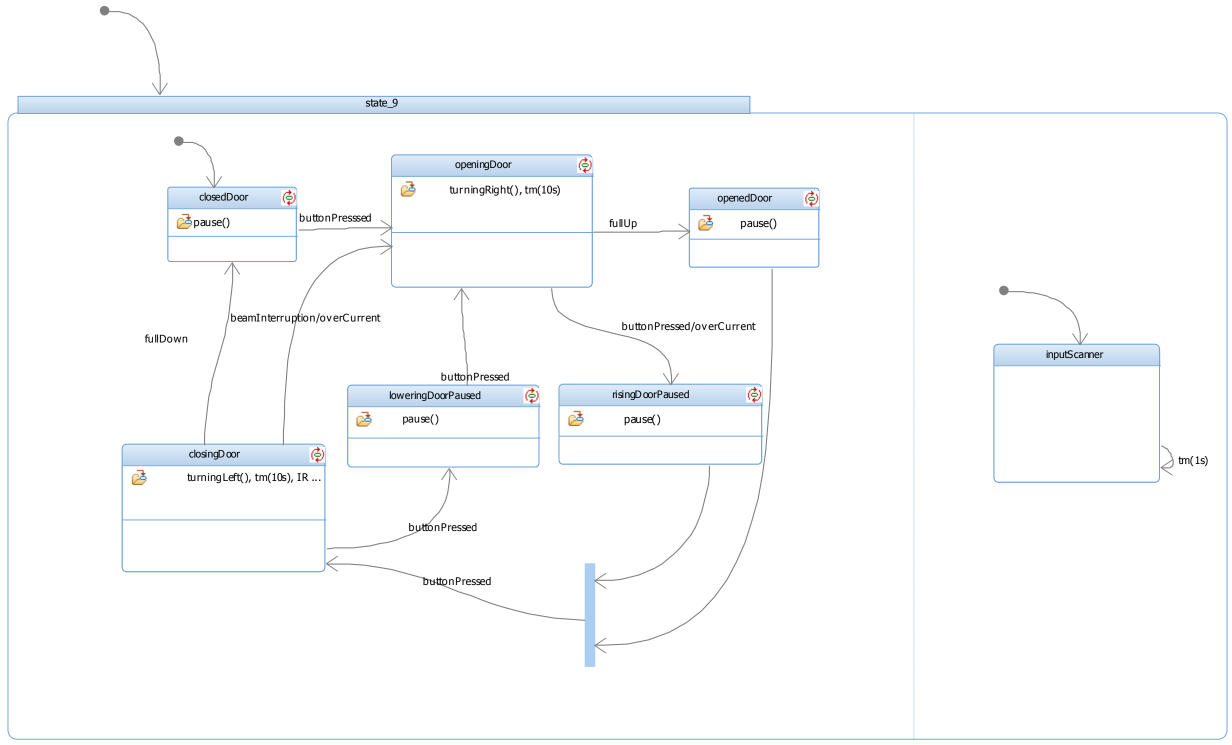
Keypress detection at closed door and ends motor stopped when door is fully opened.



This sequence diagram represents the interaction between the objects in the project. This specific sequence represents the transition of the state-machine from the closed state to open. The input scanner is activated by an event occurred after pressing the remote button. This event is handled by the garage door controller, which also runs the motorUp() action from the motor. After 10 seconds the FPGA will activate the fullUp() signal (Open door) that will be identified by the input scanner and handled by the garageDoorController, which will also tell the motor to stop spinning.

4. Statecharts





5. Discussion of the implementation of the state-chart

The state-charts for this stage of the project are the same as in the previous project. This is because the same classes with the same transitions were used. The difference between the software simulation code and the hardware code is minimal, basically the IO ports. The modified was inside the methods. A variable simulation_type (hadware or software) would determine how the code should behave in that state, but the transitions between states remained the same.

6. General comments of rhapsody.

   For this part of the project, I understood better the development of sequence diagrams using the rhapsody tools. Since the implementation of both programs is very similar, the behavior of both state-charts is also very similar.