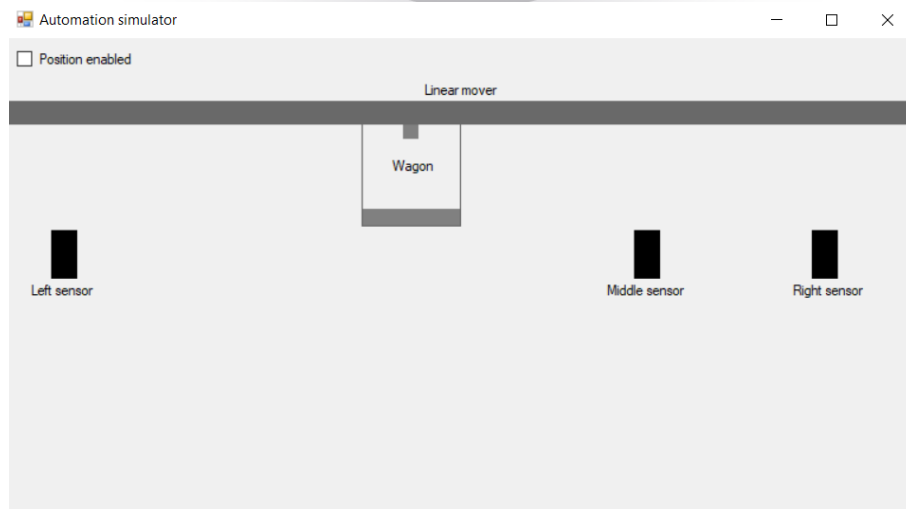


5 Automation engineering

Control an automation system according to an informal specification.

5.1 Open the solution 05_AutomationSimulator

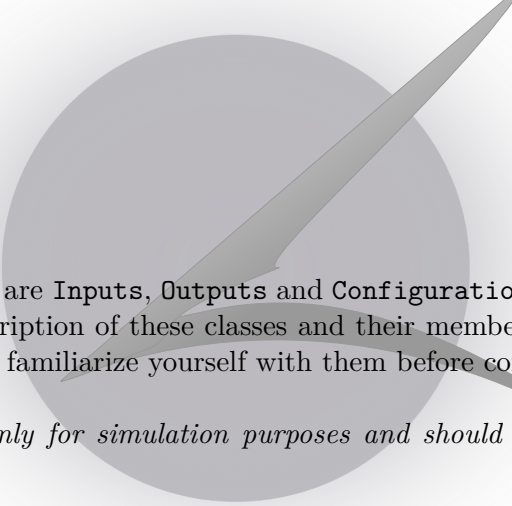


The project consists of a physical 2D process simulation and a visualization. The components are

- There is a wagon that is moving left and right. It has a limited acceleration a which is given through the configuration.
- There are three proximity sensors: Left, middle and right. The proximity sensor triggers if the wagon is (at least partially) above it. The sensors are simulated to be ideal, i.e. they trigger immediately after detecting an object.
 - The proximity sensors are shown green when they are triggered.
- The user of the system can enable or disable positioning using a switch (which is shown as the checkbox on the top left of the screen).

The simulation and visualization of the process are already implemented, but currently, the wagon will not move. The missing part is the **Controller** class, which has one method `Update()` and is intended to control the movement of the wagon:

```
1 public class Controller
2 {
3     public static Outputs Update(Inputs inputs)
4     {
5         // TODO!
6     }
7 }
```



The other relevant classes are **Inputs**, **Outputs** and **Configuration** defined in the `Contracts.dll`. The exact description of these classes and their members is written as comments in the code. Please familiarize yourself with them before continuing.

The rest of the code is only for simulation purposes and should be treated as a black box!

5.2 Design a finite state machine for the wagon control

The intended behaviour of the system is given as the following informal specification:

1. General: If the *position enabled* switch is disabled, the wagon should not move at all.
2. General: The wagon should never move past the left and right proximity sensors.
3. When the *position enabled* switch is enabled, the wagon should start moving fast towards the right until the middle proximity sensor is triggered. After that, it should continue moving slowly to the right until the right proximity sensor is reached.
4. Once the right proximity sensor is reached, the wagon should stop and position itself precisely in the center point between the middle and right proximity sensors - *as precisely as possible*. Since no information about the current speed and position of the wagon is known, the positioning needs to be timing-based.

Design a finite state machine for the wagon control using only the variables defined in the three classes **Inputs**, **Outputs** and **Configuration**.

5.3 Implement `Controller.Update()`

Please make changes only to the class `Controller`. Please treat the rest of the simulation as black box. Do not retrieve the locations or sizes of the controls from the main window, for example.