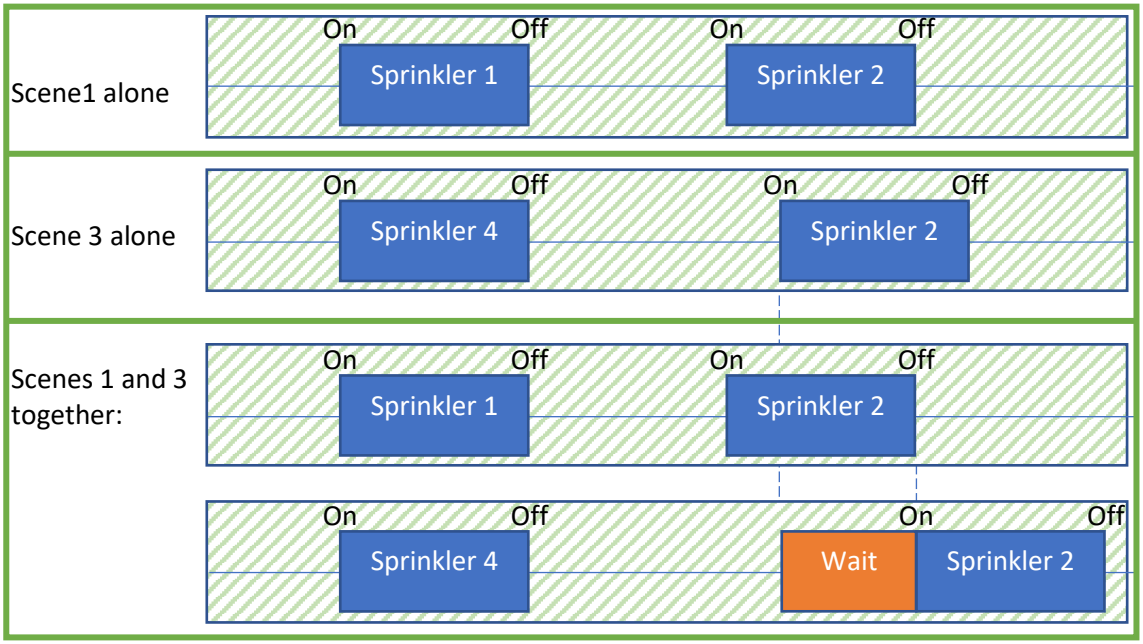
ES T3, Applying Mutexes

Extend the program of last week:

1. Make sure that different Scenes do not interfere with each other when they use the same Sprinkler: if 2 Scenes use the same Sprinkler, then the second one must wait until the first one is finished.
   For example: when Scene 1 did already switch Sprinkler 2 on, and Scene 3 also wants to switch Sprinkler 2 on, then Scene 3 must wait until Scene 1 switches Sprinkler 2 off.

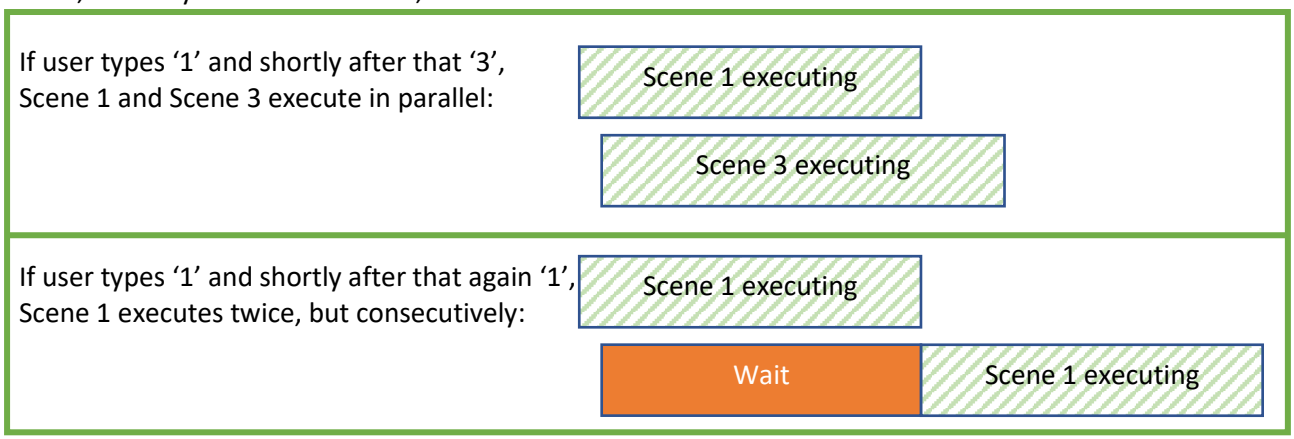


   Implement this by adding one Mutex per Sprinkler.
   Before switching a Sprinkler on, acquire its Mutex; after switching a Sprinkler off, release its Mutex.

2. <u>Optional</u>:
   Adapt your program such that Sprinklers are placed in groups.
   For example: every 4 sprinklers are in the same group:
   - 1, 2, 3, 4 in group A
   - 5, 6, 7, 8 in group B
   - Etc

   Then change the code, so it also never happens that 2 Sprinklers in the same group are switched on at the same time.
   For example: when Scene 1 switches Sprinkler 2 on, and Scene 3 wants to switch Sprinkler 4 on (which are both in group A), then Scene 3 must wait until Scene 1 switches Sprinkler 2 off.

3. Now make sure that a Scene can not execute twice at the same time.
   For example: if the user types '1', and shortly after that '3', then Scene 1 and Scene 3 will execute in parallel (at the same time), because they each have their own thread;
   but if the user types '1', and shortly after that again '1', then Scene 1 should start executing once, and only when it is finished, execute for the second time.



   Implement this by adding one Mutex per Scene.
   Before a thread executes the first step of a Scene, it must acquire the Mutex of that Scene, and after the last step of a Scene is executed, it must release that Mutex.

4. Optional:
   Make it such that the user can define a new Scene, or redefine an existing Scene, from the laptop, by sending the new operations to the Nucleo over the serial port.
   At first, don't worry about possible synchronization issues.
   This can be implemented in many different ways, you are free to choose one.
   As an example (but any other message format is also allowed): if you send this string:
   "sc3: sp1 on, wait 5, sp1 off, sp2 on, wait 5, sp2 off"
   then:
   - If there is no Scene yet with number 3, then a new Scene is defined that gets the number 3, and that contains the mentioned operations
     or
   - if there is already a Scene with number 3, then that Scene is redefined;
     the old operations are thrown away, and are replaced by the mentioned operations.

   The new functionality that you added in this step will require some additional synchronization, because a Scene should not be redefined while it is being executed.
   Implement and test this extra synchronization.

Extend your document with the following:
- Explain how you solved the synchronization for step 1 and 3 (and optionally that for step 2).
- If relevant: how did you design step 4: what message(s) exactly must be sent from the laptop to the Nucleo to define or redefine a Scene?
- What extra synchronization was required in step 4, and how did you implement that?
- How did you test this extra functionality?
- Any special problems you had, and how you solved them.
- Optional: describe any special things that you added to the application, which were not mentioned above.