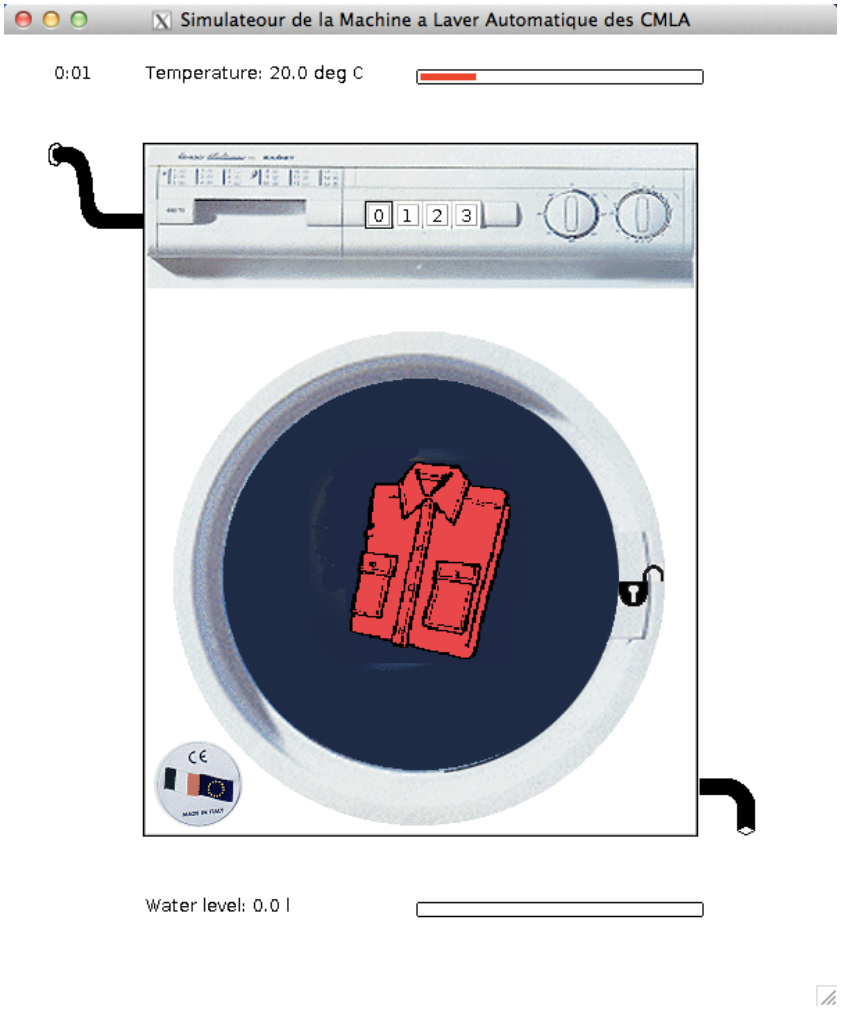
1 Lab 3 Instruction



Read the entire document before starting! It is also a good idea to read the lab preparation exercise.

You will implement the control system for a washing machine. The system consists of two kinds of activities; periodic (for controlling the hardware) and transactional (for implementing washing programs). You will use the mailbox synchronization primitive to let washing programs steer the hardware controllers. A hardware simulation allow you to test your controller software prior to deployment on the real hardware.

1.1 Objective

After the lab you should know the following terms:

- Mailboxes
- Periodic and transactional activities

and be able to

- Share data between threads using asynchronous communication
- Design and implement a small embedded system from a written specification

1.2 Preparation (before the lab)

It is generally a good idea to start programming the exercise before the lab occasion. *Before* you start programming you need to have a design of the system. This is what you prepare at the exercise session before the lab. The design should answer the following questions:

- Which concurrent activities (i.e., threads) do you need? Think about which inputs the system should handle, and how it should handle them.
- Is there any thread that is more time-critical than the rest (and should be given a higher priority)?
- Is it appropriate to introduce more threads to accomplish a clearer and more understandable design?
- Will the CMLA requirements be easy to verify (by quickly inspecting the code) in your implementation?
- Which threads need to communicate? Draw a simple diagram to show your design. What information (which classes) should be communicated?

1.3 Getting started

The instruction assumes you are sitting at a linux school computer with a shell (terminal/console). It is preferable if you have created a home folder for the course, i.e. `mkdir eda040; cd eda040`. You need to copy the handout code to your home folder[1]:

```
cp -r /usr/local/cs/rtp/lab/washing .
```

Do not forget the end point (.). Compile the code:

```
cd washing
javac -cp .:ljrt.jar done/*.java
javac -cp .:ljrt.jar todo/*.java
```

and start the washing simulation:

```
java -cp .:ljrt.jar done.Wash
```

You should now see the washing machine simulation view. However, it is not possible to wash since the controller is not implemented (yet).

1.4 Programming

The *todo* folder contains skeletons for most of the classes that you need to implement. You should implement your controller using these classes.

1.5 Getting approved

To pass you need to show your solution to the lab supervisor. You should be able to explain what you have done and motivate why you have done it. A working Java solution free of realtime problems is compulsory. Common problems are race conditions (the washing machine burns up, etc.), data corruption and excessive use of CPU.

[1] At home you can use sftp to get the files:
scp -r username@login.student.lth.se:/usr/local/cs/rtp/lab/washing .

1.6 Miscellaneous

- *At home* You should be able to complete this exercise on your own computer. You need to download the code using a sftp client such as filezilla.
- In this exercise you *must* use mailboxes in your solution. Other primitives, such as semaphores and monitors are not allowed.