EURON 2003 TeleLab
on Agent-based Mobile Robots

Part I: Running the Agent Controller

Step 1: Open the system

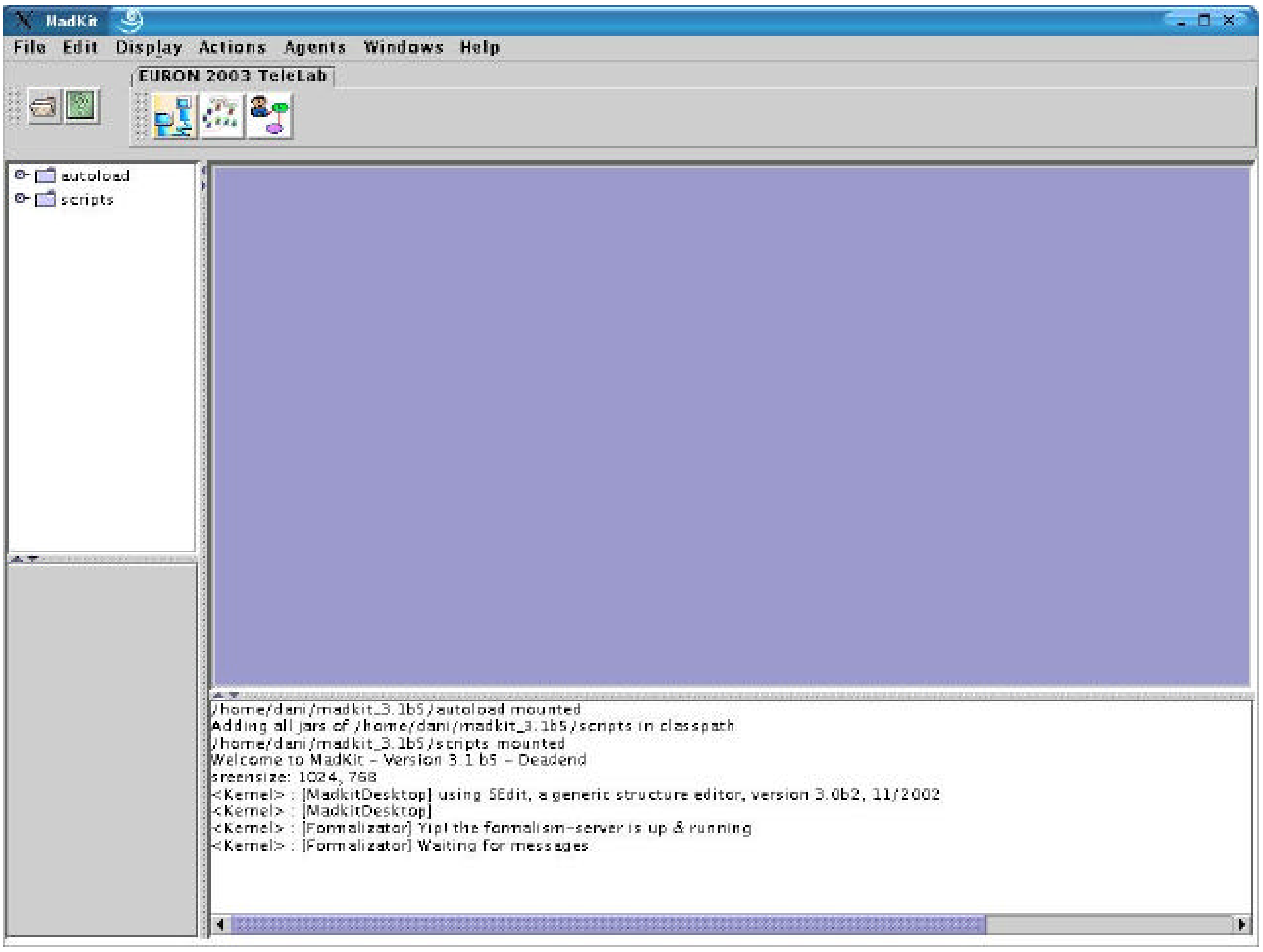
First to all log in to the system, as user **usuario** and password **practicas**
Then open the Madkit Enviroment, as follows:

1. Open a terminal.
2. Enter the MadKit directory: *cd madkit_3.1b5*
3. Lauch MadKit: *desktop.sh &*

The Graphical enviroment takes a few seconds to load. Please wait.

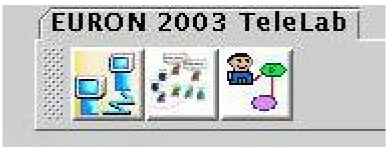
Step 2: Launching the Communicator Agent

When the graphical enviroment is loaded you can rearrange the frames. You can see 4 diferents frames or windows, as indicated in the following picture:



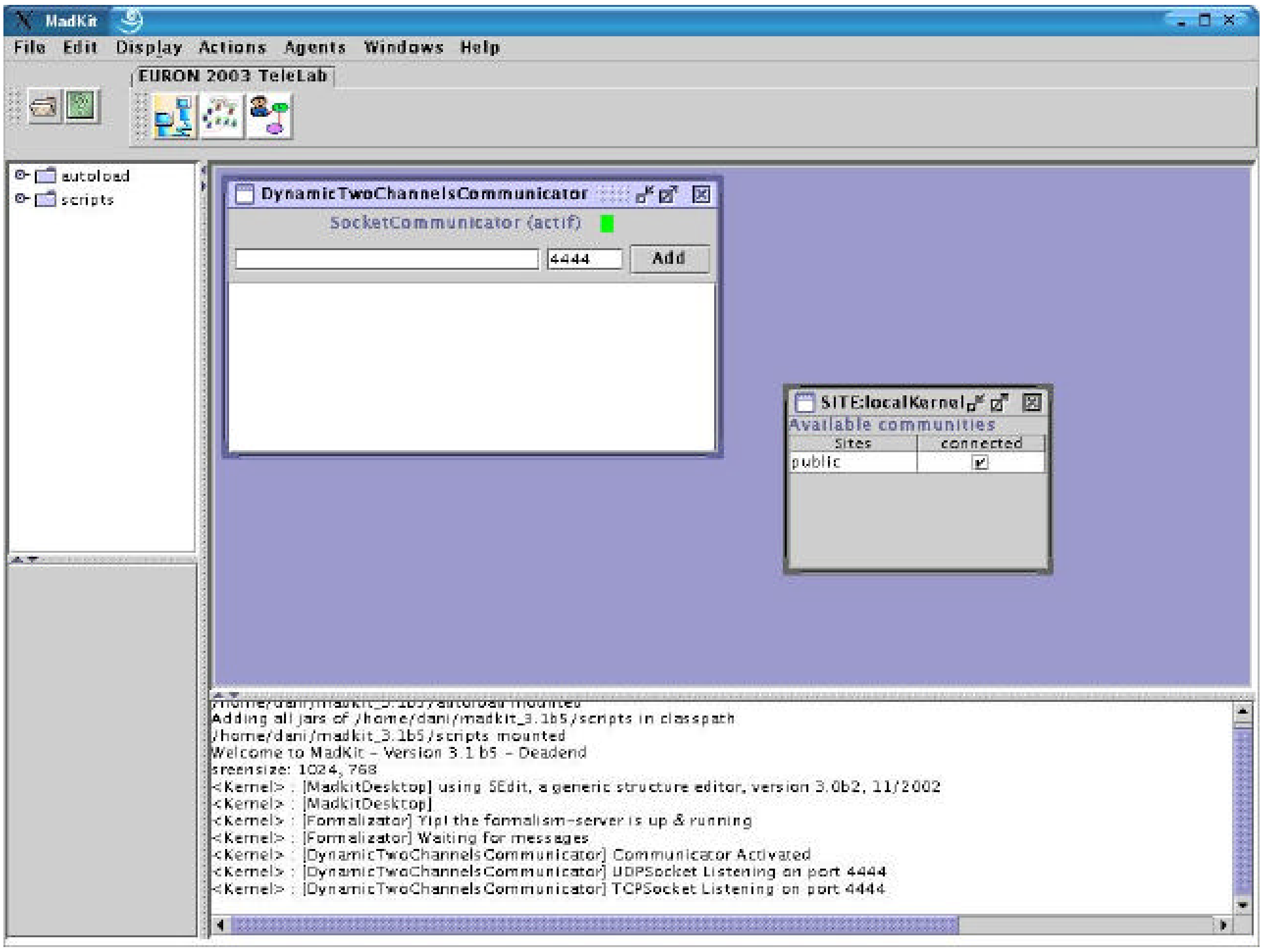
```
/home/dani/madkit_3.1b5/autoload mounted
Adding all jars of /home/dani/madkit_3.1b5/scripts in classpath
/home/dani/madkit_3.1b5/scripts mounted
Welcome to MadKit - Version 3.1 b5 - Deadend
sreensize: 1024, 768
<Kernel> : [MadkitDesktop] using SEdit, a generic structure editor, version 3.0b2, 11/2002
<Kernel> : [MadkitDesktop]
<Kernel> : [Formalizator] Yip! the formalism-server is up & running
<Kernel> : [Formalizator] Waiting for messages
```

You can see 3 buttons in the top side, at the left.



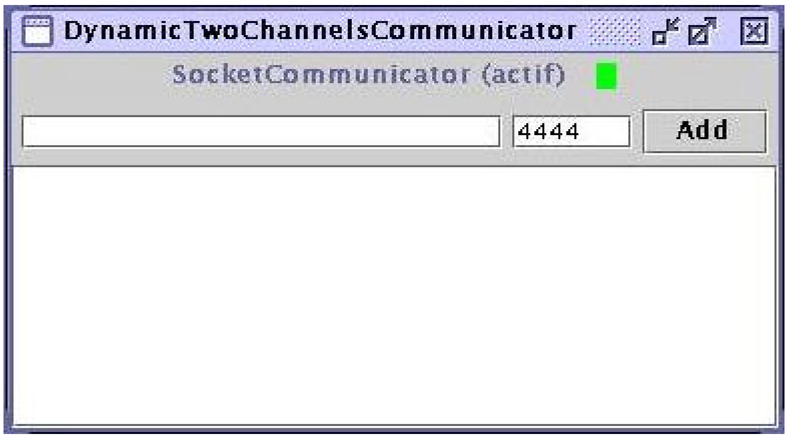
Press the first button, **Communicator**

Now, two windows must be open in the workspace:



```
/home/dani/madkit_3.1b5/autoload mounted
Adding all jars of /home/dani/madkit_3.1b5/scripts in classpath
/home/dani/madkit_3.1b5/scripts mounted
Welcome to MadKit - Version 3.1 b5 - Deadend
sreensize: 1024, 768
<Kernel> : [MadkitDesktop] using SEdit, a generic structure editor, version 3.0b2, 11/2002
<Kernel> : [MadkitDesktop]
<Kernel> : [Formalizator] Yip! the formalism-server is up & running
<Kernel> : [Formalizator] Waiting for messages
<Kernel> : [DynamicTwoChannelsCommunicator] Communicator Activated
<Kernel> : [DynamicTwoChannelsCommunicator] UDPSocket Listening on port 4444
<Kernel> : [DynamicTwoChannelsCommunicator] TCPSocket Listening on port 4444
```

You can see a window labeled as **Dynamic Two Channels Communicator**



in this window add the IP of your server, following the next table:

Name	IP

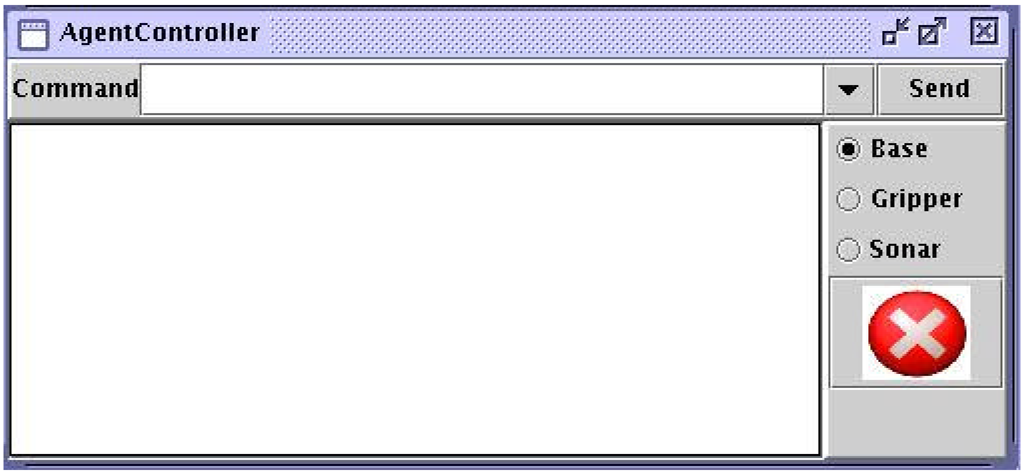
Dopey	150.128.49.172
Sleepy	150.128.49.173
Bashful	150.128.49.174

If you have any problem in this Step, please communicate it to the Lab teacher.

Step 3: Executing the Robot's Agent Controller

Now, launch the Agent Controller pressing its button

A new window is opened in the workspace.



Now you can select differents commands using the menu, and the robot must execute the diferents orders. If anything goes wrong, press the stop button. Enjoy!

Part II: Creating a Simple Agent

Step 1: writing the agent's source code

VERY IMPORTANT: if Madkit is running, please quit from the application.

Go into the *acromovi* directory and edit the file *template.java*. This file contains the basic skeleton of a Madkit agent:

```
public class template extends Agent{
  public void activate(){
    requestRole("Seven Dwarfs", "Sleepy", "template", null);
    println("Created group and role.");
  }
  public void live(){
    /* Here goes your code */
  }
  public void end(){
    leaveRole("Seven Dwarfs", "Sleepy", "template");
  }
}
```

Prior to adding any code, you should learn how the agent sends commands to the elements of the robot. This is done by means of the function **sendOrder**, which sends an string message (the order) to the specified element of the robot:

```
String[] sendOrder(String element, String command)
```

The result of this function is an array with the data returned (if any) by the robot element, either the base, the sonar, or the gripper.

An example of use of this function is:

```
String[] tmp = sendOrder("Sonar", "(getPose)");
```

Such example returns an array of three elements (x, y, heading) of the actual position and orientation of the robot.

Now, using the functions listed in the reference, you have to write an agent that drives the robot to follow a square trajectory.

Once your code is ready, you should save the file and quit from the editor.

Step 2: Compiling and running the agent

In order to compile your agent's code, you should simply run **make** on the makefile located in the same directory.

Next, you should start Madkit, and run the Communicator agent, as previously, making the connection with your robot.

Now, press the third button ; your program will execute, and hopefully move the robot in the planned trajectory (in case of failure, go back to editing the source).

Part III: creating a *wandering* agent

Follow the same steps of the previous part to create this agent. You only need to edit the **live** method in the agent's template. The goal of this agent is to drive the robot in a random walk trhough its environment without colliding with the surrounding objects. The robot should move in the direction towards free space, or the most distant obstacle.

Compiling and running the agent is done as explaind previously too.

Important: some useful functions are implemented in the file *template.java*:

- **sendOrder(element, command)**: sends the specified command to the robot element.
- **stringToDouble(string_array)**: converts the array of strings into an array of doubles.
- **mayor(double_array)**: returns the maximum element of the array.
- **calculate_dir(num_sonar)**: returns the direction corresponding to the specified sonar.
