Working with the robots

Now you know how to write robot software, lets discuss how to work with the hardware. We are using two types of robots, Jackal and the rosbots Bobo and Coco. Their operation will be slightly different.

Jackal

When working with Jackal you will be provided with a laptop, hereafter called the robot-laptop. This laptop is already connected to the robot via a wifi connection. You will use this laptop to run your software on the robot.

Hero (just in case)

starting Hero

To start the robot long press the power button, after about three seconds you will hear the fans starting up. Once you can see the battery level on the screen of the robot (NOT a touchscreen!) the robot is properly started. You can now start the hardware by releasing the emergency stop button. The robot will assume its default pose.

To stop the robot simply press the emergency stop button. This will stop the hardware (but not your software!). When the emergency button is pressed you can push the robot around. Pushing the robot requires some force but please do be careful. Releasing the emergency stop will cause the hardware to start up again. Do not attempt to move the robot by hand when the emergency button is released.

Bobo and Coco

You can connect to the rosbots with your own laptop, the teaching assistants will provide you with the wifi password.

starting Bobo and Coco

When you turn on the power switch on the rosbots the computer will start up but the hardware will not start running yet. To start the hardware open a terminal, type `sshbobo` or `sshcoco` depending on your robot (More on the `ssh<robot>` commands later in this document). Then type `bobo-start` or `coco-start`. You should notice the lrf start spinning. The hardware is now starting up.

The rosbots do not have an emergency stop. But they are light enough to not be a danger. You can pick them up if you need to stop them quickly. To stop the hardware you can press ctrl+C in the terminal running bobo/coco-start.

robot core

Normally all your tools, like Rviz, teleop, etc will try to communicate with the simulator running on your laptop. We should specify that we want to listen to the signals passed on the robot. To do this you can input

```
<robot>-core
```

in a terminal. From that point on any command you do afterward will try to communicate with that robot.

rviz

To visualize the LRF measurements open a terminal on your laptop and input `hero-rviz`, `bobo-rviz` or `coco-rviz` depending on your robot.

Teleop

Instead of pushing the robot by hand you can control the robot remotely using your keyboard. On the robot-laptop open a terminal and input.

```
mrc-teleop
```

and you will be able to move the robot forward and backward with 'w' and 's', you can rotate the robot using 'a' and 'd' and you can stop movement with any other key.

ssh

Controlling the robot remotely is all well and good. But we want to make changes to the software on the robot itself.

To connect to the robot open a terminal on the robot-laptop and input `sshhero` / `sshbobo` / `sshcoco`. You will notice the green text changing to `mrc@hero1` (or something else on the rosbots). This indicates that you are now operating a terminal on the computer inside the robot. You will find a folder called group_repos here where you can use git pull to get your code. Remember to build your code as only source files will be on git. In order to ssh into the rosbots you might have to provide the password of the rosbot. The password is provided by the student assistents.

⚠ Warning

Note that since you are working on the computer in the robot, commands like `<ROBOT>-rviz` and `gedit` will try and fail to open a window. The only operations you can perform here are ones you can do from the terminal, including the git commands!

Exercise

Don't crash

You now know how to work with the robot. For your first session, we suggest you start by experimenting with the robot. **document the answers to these questions on your wiki**

1. On the robot-laptop open rviz. Observe the laser data. How much noise is there on the laser? What objects can be seen by the laser? Which objects cannot? How do your own legs look like to the robot.

Go to your folder on the robot and pull your software.

2. Take your example of don't crash and test it on the robot. Does it work like in simulation? Why? Why not? (choose the most promising version of don't crash that you have.)
3. If necessary make changes to your code and test again. What changes did you have to make? Why? Was this something you could have found in simulation?
4. Take a video of the working robot and post it on your wiki.