# Robot Practical Course Bachelor

## Assignment #1

In this assignment you will setup the software framework ROS which will be used throughout this course. You will get to know the fundamentals of the framework using the tutorial-simulator "turtlesim". The goal is to draw a "Haus vom Nikolaus" (an old german drawing puzzle for kids) in the simulator.

**NOTE: We will use the ROS distribution "noetic" in this course. So always replace "<ROSVERSION>" with "noetic"!!!**
**LaTeX renders the tilde (~) as a different character. Do not copy it into your shell.**

**Task 1.1 Set up ROS environment:** Software for ROS is usually developed in "workspaces". Because you will develop ROS packages in the next tasks, first of all you have to setup such a workspace for yourself.

**1.1.1:** Boot the lab computer to Ubuntu and login using your normal account (e.g. `6musterm`).

**1.1.2:** Create a folder you can use as your ROS workspace. In a terminal, type:

```
mkdir -p ~/pc22_ws/src
```

**1.1.3:** Next, turn this folder into a workspace. To do so, first you have to load the ROS environment that is already installed on the lab computers:

```
source /homeL/demo/turtlebot_ws/devel/setup.bash
```

Now you can initialize the workspace:

```
cd ~/pc22_ws
catkin build
```

**1.1.4:** By now, you could already load your new workspace and work with it. However, to ease your development, you should instead make your account load the workspace for you whenever you open a new terminal. To achieve this, add the following line to the file `~/.bashrc` .
**Do not just execute it in the terminal**

```
source ~/pc22_ws/devel/setup.bash
```

**1.1.5:** Finally restart your shell to load the workspace in your current terminal:

```
exec bash
```

**Task 1.2 Start turtlesim:** In this section you will learn how to start ROS and ROS nodes. You will teleoperate a turtle in turtlesim and get to know basic ROS commands.

**1.2.1:** Open 4 terminals.
Start a roscore in the 1st one:

```
roscore
```

Start the turtlesim node in the 2nd:

```
rosrun turtlesim turtlesim_node
```

This brings up the turtlesim environment with one turtle in it. To teleoperate this turtle you can start another node to send commands for the turtle in the 3rd terminal:

```
rosrun turtlesim turtle_teleop_key
```

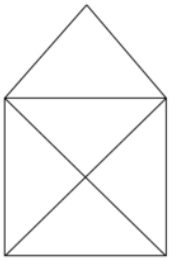
turtlesim should be started and you should be able to move the turtle around by using the arrow keys.
Now try some of these commands in the 4th terminal to learn about the environment you just started:

```
rosnode list
rostopic list
rostopic info <TOPICNAME>
rostopic echo <TOPICNAME>
rosservice list
rosrun rqt_graph rqt_graph
```

**Task 1.3 Explain ROS:** Explain first to each other and then to a supervisor the basic concepts of ROS. You should name:

- Nodes
- Communication
  - Messages
  - Topics
  - Services
  - Actions
- ROScore
- Packages

**Task 1.4 Create your first ROS node:** Write a node that moves the turtle around, such that it draws a ”Haus vom Nikolaus”. It looks like this and has to be drawn in one move.



**1.4.1:** Create a ROS package with your group’s name in your workspace. For this task, look up this tutorial:

`http://wiki.ros.org/ROS/Tutorials/CreatingPackage`

Notice that ‘catkin‘ is the main build-system for ROS that is used to maintain workspaces. Dependencies can be changed later on but it is easier to specify them from the start. Your program will depend on the ‘turtlesim‘ and the ‘geometry_msgs‘ packages.

**1.4.2:** Look up the ROS tutorial for a simple publisher. There is one for C/C++ and one for Python.
**We strongly recommend using Python**.
**You have to make sure you look at the tutorial for the right version of ROS**.

C/C++: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29`
Python: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29`

**1.4.3:** Now write a node (either in C++ or Python) that publishes messages that make the turtle draw a ”Haus vom Nikolaus”. In this tutorial you can look up which topic the turtle uses:

`http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics`

Every package for ROS has to be build at least once in order to make it known to the workspace. This has to be done even if you implement the node within the package in python. Afterwards you can start your node with this command (just as you started the nodes in the previous task):

```
rosrun <ROS_PACKAGE> <THE_PROGRAMM>
```

**1.4.4:** Read up on the Services available by Turtlesim. Try calling some of turtlesim’s Services through the command line.

```
rosservice call <SERVICE_NAME> <ARGS>
```