Robotics Practice 1 Task Sheet

This handout is available at: http://shtech.org/course/ist/17s/homework/robotics/lab1/practice1.pdf

Do Tasks 1 + 2 + 3 + 10 + 11 + (two of: 4, 5, 6, 7, 8, 9). It is sufficient if one person per group completes a task – it counts towards the whole group. You can work in parallel (different persons working at different tasks at the same time) or sequentially (all persons working together on each problem). Once you have completed a task call the TA or Prof. such that we can check and note your progress.

Tips: Ubuntu 16.04 and ROS kinetic have been tested working. We recommend you to use that version in a real machine (no virtual machine) for the robotic lab.

Task 1: Startup the robot

You need to complete the installation of the kobuki on your laptop. Only once run:

rosrun kobuki_ftdi create_udev_rules

Connect the USB cable to the robot and your laptop and run:

roslaunch kobuki_node robot_with_tf.launch

You should hear some sound. You can lookup the tutorials for kobuki here: http://wiki.ros.org/kobuki/Tutorials

Task 2: Do KeyOp

Download http://shtech.org/course/ist/17s/homework/robotics/lab1/keyop.py
Save it to the “Robotics Lab Code” homework repository on the gradebot in a folder called “test”.
Execute the keyboard control using:
python keyop.py
Place your laptop on the robot and drive around with the robot.
Take a look at the code and play with it. Increase the maximum speed to 1.2!

Task 3: Use the joystick

Install the required package:

sudo apt-get install ros-kinetic-joy

Download http://shtech.org/course/ist/17s/homework/robotics/lab1/joy.py
Save it to the “Robotics Lab Code” homework repository on the gradebot in a folder called “test”.
Execute the joystick control by running (each in its own console):
rosrun joy joy_node
python joy.py
Place your laptop on the robot and drive around with the robot.
Take a look at the code and play with it.

Task 4: Set LED status using rosmsg pub

Find out what topics are made available by the robot software:
rostopic list
Set the status of the leds. Use (you can use tab completing (double press “Tab”) !):
rostopic pub <topic> <messageType> <data>
What else can you play around with?

Task 5: Try remote control via ssh

Beware: you are about to install a software that grants others access to your laptop! Only use it if you have a secure password! You can remove the software later (**sudo apt-get remove openssh-server**)

sudo apt-get install openssh-server

Connect your laptop to the wireless network. Take note of your IP address: (run: **ifconfig**). Place your laptop on the robot. From another laptop (which is connected to the wireless, too) run:
ssh user@ip
You need to provide your password. Now you can start the teleop node from here.

Task 6: Try remote control via ROS

Start the kobuki node on the laptop on the turtlebot (see Task 1). Also note the IP address of the laptop. Connect to the network.
Use a second laptop running Ubuntu and ROS (note: this will only work if both laptops run Linux directly (or at least directly connect to the network – NAT configured VM's will not work).
On the second laptop install the joystick stuff (Task 3). Open a new console on run:
export ROS_MASTER_URI=http://ip:11311/ (exchange ip with the ip of the turtlebot-laptop)
If there is a roscore running on the turtlebot-laptop (e.g. because you used **roslaunch** from Task 1) you can test on the operator laptop is everything is all right by executing:
rostopic list
If you see a list everything is all right, if it complains a missing ros master you have to debug some more.
If the above was successful you can run now the joystick node (with the joystick connected to the operator lapotp) on the laptop. The start joy.py from Task 3 on either laptop. Why does it work on either Laptop? How does the data flow?

Task 7: Try and play with shape

Download http://shtech.org/course/ist/17s/homework/robotics/lab1/shape.py
Save it to the "Robotics Lab Code" homework repository on the gradebot in a folder called "test".
Execute the shape program using:
python shape.py
Press 1 or 2 to let the robot drive a pre-programmed path. Play around with it and program your own shape!

Task 8: Modify KeyOp to measure the distance traveled

Copy keyop.py to measure.py. Edit measure.py. Subscribe to the odometry on the topic **/odom** . The message type is **nav_msgs/Odometry**. See http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29 for information on that. Use the position information from the odometry to find out how much you drove since the last odometry message. Add up those distances and display them! Add measure.py to git!

Task 9: Start Kinect and visualize in rviz

Install openni:
sudo apt-get install ros-kinetic-openni-camera ros-kinetic-openni-launch

Install driver:
wget http://robotics.shanghaitech.edu.cn/static/TMP/avin2-SensorKinect-15f1975.tar.gz
tar xfvz avin2-SensorKinect-15f1975.tar.gz
cd avin2-SensorKinect-15f1975/Bin
tar -xjf SensorKinect093-Bin-Linux-x64-v5.1.2.1.tar.bz2
cd Sensor-Bin-Linux-x64-v5.1.2.1
sudo ./install.sh

Start the camera:
roslaunch openni_launch openni.launch

See details in: http://answers.ros.org/question/60562/ubuntu-12042-and-openni_launch-not-detecting-kinect-after-update/
Connect the kinect to power (if not already done) and the usb to your laptop. Show the Kinect data in rviz:
rosrun rviz rviz

Task 10: Communicate with the photo sensor

In the Electronics Lab, you have learned to use the Arduino to obtain sensor data and visualize it on the computer, you have written some code in Arduino IDE. Now, we also want to get the sensor data on ROS with the help of Arduino. To do that, we need rosserial(http://wiki.ros.org/rosserial), especially rosserial_arduino and rosserial_python.

Arduino IDE Setup

For Ubuntu just use the command “sudo apt-get install arduino” to install Arduino IDE. It will configure the serial permission automatically when you first start it.

Others should download from here https://www.arduino.cc/en/Main/Software and install it. You may need manually add your user to dialout group by command. '**sudo adduser your-username dialout**' Don't know your username? Run '**whoami**' in terminal.

Once you have installed Arduino IDE, to have it work with ROS, we need rosserial library in Aruino IDE.
For Ubuntu just use the following command:

```
sudo apt-get install ros-kinetic-rosserial-arduino
sudo apt-get install ros-kinetic-rosserial
cd ~/sketchbook/libraries
rm -rf ros_lib
rosrun rosserial_arduino make_libraries.py .
```

Others may look here for help http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup.

What's more, we also have a timer on Arduino using another library named FlexiTimer2 (http://playground.arduino.cc/Main/FlexiTimer2). Download the zip file and rename it as "FlexiTimer2.zip". In the Arduino IDE, navigate to Sketch > Include Library. At the top of the drop down list, select the option to "Add .ZIP Library", then select FlexiTimer2.zip.

After installing rosserial and FlexiTimer2, in the Arduino IDE, navigate to Sketch > Include Library, you will see them at the bottom of the lists. If so, then you can compile this Arduino program (save it to the gradebot “Robotics Lab Code” homework in a folder called “race”): http://shtech.org/course/ist/17s/homework/robotics/lab1/AnalogReadSerial_ros.ino

You are welcome to edit the Arduino program according to your needs.

rosserial_python

In the last step, we setup everything on the arduino-side, but actually you don't really publish a topic on your computer, we need another node to help us to do that. Connect the Arduino with your computer. In the terminal, we run

rosrun rosserial_python serial_node.py /dev/ttyACMx

'**/dev/ttyACMx**' is the label of Arduino on the computer, x is a number, e.g. 0,1,2,... . If you want to make sure the number, run

ls /dev/ttyACM*

while Arduino is connected to your computer.

After the above two steps, run '**rostopic list**' in a new terminal, you should see the topic named "**ir_data**" which we published on Arduino. Use '**rostopic echo**' and '**rostopic hz**' to display the contents and frequency of the data.

Task 11: Download the race template program

Download the template for the race program: http://shtech.org/course/ist/17s/homework/robotics/lab1/race_ir.py
Save it to the “Code for Practice” homework repository on the gradebot in a folder called “race”.
In the program print the messages received.

As a homework for the 2nd lab your group has to implement three **different functions**:

1. Decode the two signal values from the Int32. Take a look at the Arduino program how they are encoded.
2. Determine the position of the robot with respect to the cones.
3. Determine the control output send to the robot, based on the position within the path.

You may also implement the control program differently, but those three steps are recommended.
Use the bagfile (http://shtech.org/course/ist/17s/homework/robotics/lab1/robot2017.bag) to test your code without a robot!