ROS: Creating a ROS driver for ultrasonic ranging module

The objective of this lab is to learn how to create a ROS driver for a sensor device. In this lab, you will create a ROS driver for a ultrasonic distance sensor connected to Arduino Nano. You will use rosserial to transfer data over serial interface and RViz to visualize the sensor data.

Setting up your computer

Open terminal window.

Clone your git repository to your home folder:

```
$ git clone <URL for your personal repository <yourname>-rtech>
```

Use `ls` to confirm that ***<yourname>-rtech*** has been downloaded to your home folder.

Brief introduction

This lab is the second of two labs that focus on creating a sensor driver in a ROS-based system.

On a general level, the pipeline for graphically visualizing sensor data is depicted in Figure 1. The physical sensor is connected to your computer through a supported communication interface. ROS driver node reads the data from that communication interface and publishes it in a proper format as ROS messages. Even though the ROS driver node can publish the data in a format that is suitable for visualization, it might be smart to create a separate node that handles visualization (i.e. keeping ROS nodes as small functional units that can be re-used as the hardware/sensor changes). For example, ROS driver node is often specific to manufacturer whereas the data visualizer node is specific to message type.

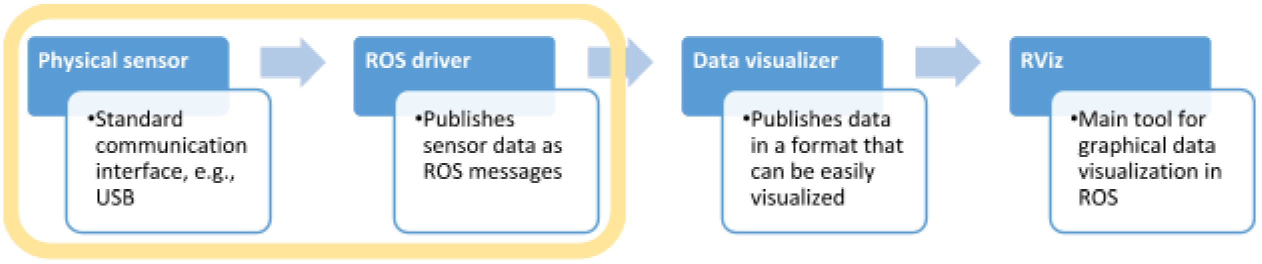
Physical sensor
•Standard communication interface, e.g., USB
ROS driver
•Publishes sensor data as ROS messages
Data visualizer
•Publishes data in a format that can be easily visualized
RViz
•Main tool for graphical data visualization in ROS

Figure 1. *The pipeline for visualizing sensor data in RViz.*

Creating a ROS driver for a sensor

In this lab you are going to:

1. Assemble ultrasonic sensor and Arduino Nano on a prototyping breadboard. Test the setup with a sample code.
2. Set up `rosserial` and test it with a sample code.
3. Create an Arduino program (ROS driver) to publish ultrasonic distance measurements as ROS messages.
4. Implement a filter in the data visualizer node created in the previous lab. Publish both raw and filtered data.

1 The datasheet for the ultrasonic distance sensor can be found here. Mount the Arduino Nano and the ultrasonic sensor on the breadboard. (**NB!** *If this is your first time using a breadboard, let the instructor know before you start.*) Use available wires to connect the sensor with the Arduino. Please note that the provided sample code defines echo pin as A4 and trigger pin as A5. Download this sample code, open it with Arduino IDE, compile it, upload it to Arduino, and verify the output using the serial monitor. Are you able make sense of the readings in the serial monitor?

If you get an error message along the lines of "permission denied", add your username to dialout group. On the classroom's laptop the command would be:

```
$ sudo usermod -a -G dialout rt-user
```

Log out and back in for the change to take effect.

2 In this lab we are using `rosserial` package to relay data from Arduino Nano to our ROS-based computer over a serial (USB) interface.

In order to install `rosserial` let us use apt-get:

```
$ sudo apt install ros-kinetic-rosserial-arduino ros-kinetic-rosserial
```

Next build the ROS libraries for Arduino:

```
$ cd ~/Arduino/libraries/
$ rosrun rosserial_arduino make_libraries.py .
```

Now when you open Arduino IDE again, navigate the menu to locate:

`File > Examples > ros_lib >` **`HelloWorld`**

Verify and upload the HelloWorld program to your Arduino Nano as is.

Next, open a terminal window and run:

```
$ rosrun rosserial_python serial_node.py /dev/ttyUSB0
```

Note: Your USB path may be different.

If everything worked out, you should now be able to see a topic named *chatter* in your list of active ROS topics. Try printing messages published to this topic on the screen. What do you see? Make sure you understand the HelloWorld code before you continue.

3 Create a new Arduino program that publishes distances from ultrasonic sensor as ROS messages.

4 Modify or add new node in the `range_visualizer` package from the previous lab to receive the messages from Arduino and apply a simple filter on the data.

End result: The ultrasonic range sensor is connected to the Arduino and the computer is receiving ranging data. Messages are being published on at least two topics, e.g., *ultrasound/raw* and *ultrasound/filtered*. RViz is used to visualize messages on either of these topics.

>>> Show the result to your lab instructor! <<<

Clean up your workspace

NB! Before you leave the lab, make sure you have pushed all the files in your catkin workspace to your git cloud service.

In terminal, `cd` to **<yourname>-rtech**

Type

```
git config user.email "youremail@example.com"
```

Type

```
git status
```

You should now see all the new and modified files in red.

Prepare the relevant files for the commit.

```
git add file_name_in_red1 file_name_in_red2
```

When you now type

```
git status
```

you should see all the added files in green. You are now ready to commit changes. Type

```
git commit -m "Insert a brief explanation"
```

Your changes have now been committed but not yet uploaded to the cloud. To upload your files, type

```
git push
```

In your web browser, **verify that all the files** have been uploaded to the **<yourname>-rtech** repository.

Delete the **<yourname>-rtech** folder and any other files you created from the lab's computer.