

Module A: Build the F1/10 Robotic Car

Part 3. ROS Tutorials

Part 4. Setup Sensors

Websites:

<http://f1tenth.org>; <http://www.jetsonhacks.com>;
<http://www.ros.org>; <https://www.hokuyo-aut.jp/> ;

Part 3. ROS Tutorials

- ❑ Make sure your host computer or Jetson has ROS installed properly
 - ❖ You may have to run `$sudo apt-get install python-rosinstall`
- ❑ Follow Basics of ROS Part 1 here: <http://fltenth.org/code.html#s1t1>
 - ❖ Pay attention to the 3 layers of ROS hierarchy: what are the contents in each of the layers in the turtlesim example?
 - ❖ What is a topic, a message, a publisher, or a subscriber in the example?
 - ❖ How do you open a new terminal using a key shortcut? How do you save the rqt graphs?
 - ❖ How to exit a ROS process without closing the terminal?
- ❑ Follow Basics of ROS Part 2 here: <http://fltenth.org/code.html#s1t12>
 - ❖ If you run into problems, find the original ROS tutorials. Pay attention to the following questions:
 - ❖ What is the catkin package? what are the contents in each of the layers in the listener and talker example?
 - ❖ What does it mean by “source an environment”?
 - ❖ What are some other tutorials on ROS website that you found helpful?



Part 3. ROS Basics <http://wiki.ros.org/ROS/Tutorials>

❑ ROS 3-Layer Hierarchy: nodes, packages, and workspaces

- ❖ A node is a program with a specific functionality that runs as a single process;
- ❖ A package is a collection of nodes and their messages and services;
- ❖ A workspace is a folder (directory) created by catkin for users to develop packages.

❑ ROS Topic, Messages, Publisher, and Subscriber

- ❖ A topic is the channel that allow messages to communicate among nodes;
- ❖ A message is the data communicated through a topic;
- ❖ A publisher is a node that outputs messages;
- ❖ A subscriber is a node that receives messages.

❑ The ROS master node: `$roscore` (may be automatically launched by `$roslaunch`)

- ❖ `roscore` is a collection of nodes and programs needed for a ROS-based system
- ❖ Must be running for ROS nodes to communicate;
- ❖ Will start up a ROS master, a ROS Parameter server, and a `rosout` logging node.

Part 3. ROS Tutorial Examples

- ❑ `$roscnode list ($roscnode info nodename)` <http://wiki.ros.org/roscnode>
- ❑ `$rostopic list ($rostopic echo topicname)` <http://wiki.ros.org/rostopic>

```
jetson-11@jetson11-desktop:~$ rostopic info /scan
```

```
Type: sensor_msgs/LaserScan
```

```
Publishers:
```

```
* /urg_node (http://jetson11-desktop:45955/)
```

```
Subscribers:
```

```
* /rostopic_10123_1581892369708 (http://jetson11-  
desktop:39843/)
```

```
jetson-11@jetson11-desktop:~$ rostopic echo /scan
```

```
header:
```

```
seq: 280612
```

```
stamp:
```

```
secs: 1581899542
```

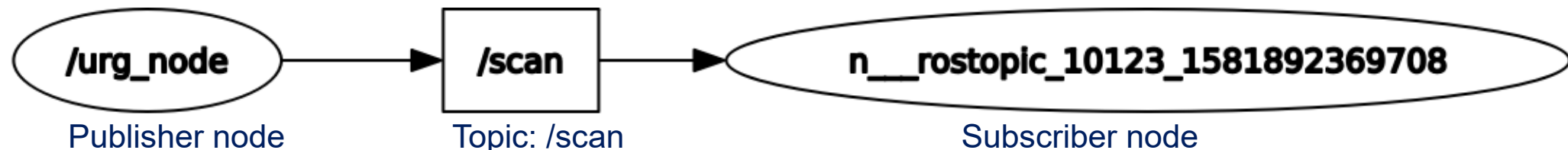
```
nsecs: 921553933
```

```
frame_id: "laser"
```

```
angle_min: -2.35619449615
```

```
angle_max: 2.35619449615
```

- ❑ `$roscrun rqt_graph rqt_graph` refresh to get update, choose Node/Topic (all) and save image

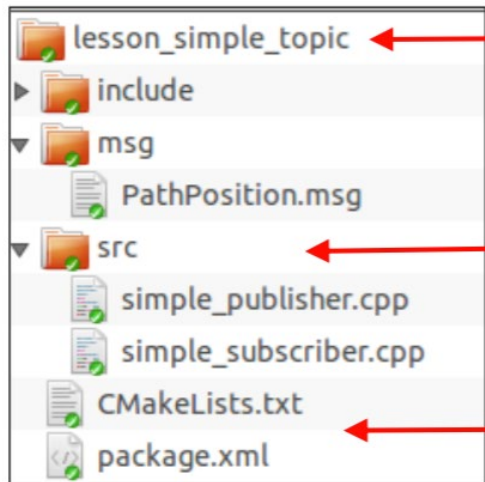


Part 3. ROS Packages <http://wiki.ros.org/Packages>

❑ A ROS package contains several required files:

- ❖ CMakeLists.txt: CMake build file (see reference <http://wiki.ros.org/catkin/CMakeLists.txt>)
- ❖ package.xml: properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages. See <http://wiki.ros.org/catkin/package.xml>

❑ ROS packages tend to follow a common structure:



- ❖ include/package_name: C++ include headers (make sure to export in the CMakeLists.txt)
- ❖ msg/: Folder containing Message (msg) types
- ❖ src/package_name/: Source files, especially Python source that are exported to other packages.
- ❖ srv/: Folder containing Service (srv) types
- ❖ scripts/: executable scripts
- ❖ CHANGELOG.rst: a changelog which can be automatically injected into binary packaging and into the wiki page for the package
- ❖ Required files: CmakeLists.txt and package.xml

❑ ROS packages are created by hand or by tools like `catkin_create_pkg`

❑ Metapackages: grouped multiple packages as a single logical package



Part 3. ROS Catkin Workspace

- ❑ Catkin workspace: contains up to four different spaces
 - ❖ Source space: source code of the package
 - ❖ Build space: where CMAKE is invoked to build the package;
 - ❖ Development (Devel) space: contains built targets prior to being installed;
 - ❖ Install space: targets are installed into the install space by `make install`;
 - ❖ Result space: refer to either a devel space or an install space
- ❑ Creating a catkin workspace: http://wiki.ros.org/catkin/Tutorials/create_a_workspace

Typical work flow with `catkin_make`:

- ❖ Source the melodic environment space: `$source /opt/ros/melodic/setup.bash`
- ❖ Create a catkin workspace: `$mkdir -p ~/myWSname_ws/src`
`$cd ~/myWSname_ws/`
`$catkin_make`
- ❖ Create install space: `$catkin_make install;`

Part 4: Setup Lidar Sensors

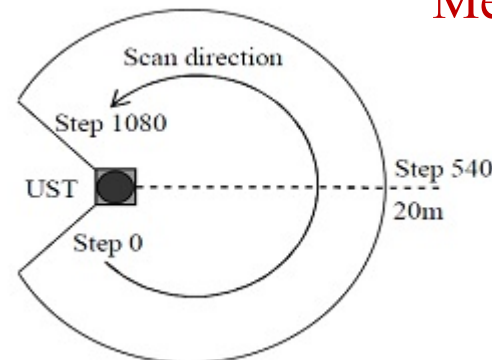
❑ 4.1 Hokuyo UST-10LX

- ❖ Ethernet cable (TCP/IP) connect to Orbitty or host computer
- ❖ JST-SH 1mm 6 pin connector to power 12V



Size: W50xD50xH70 mm,
Light Weight: 130 g
Detection range: 10 m
Measurement steps: 1081

Measurement steps 1081
Detection angle 270°
Angular resolution 0.25°



Scan speed: 25 ms (~ motor speed 2400 rpm)

❑ 4.2 Hokuyo UTM-30XL

- ❖ USB cable connect to Orbitty hub or host computer
- ❖ JST-SH 2mm 4 pin connector to power 12V

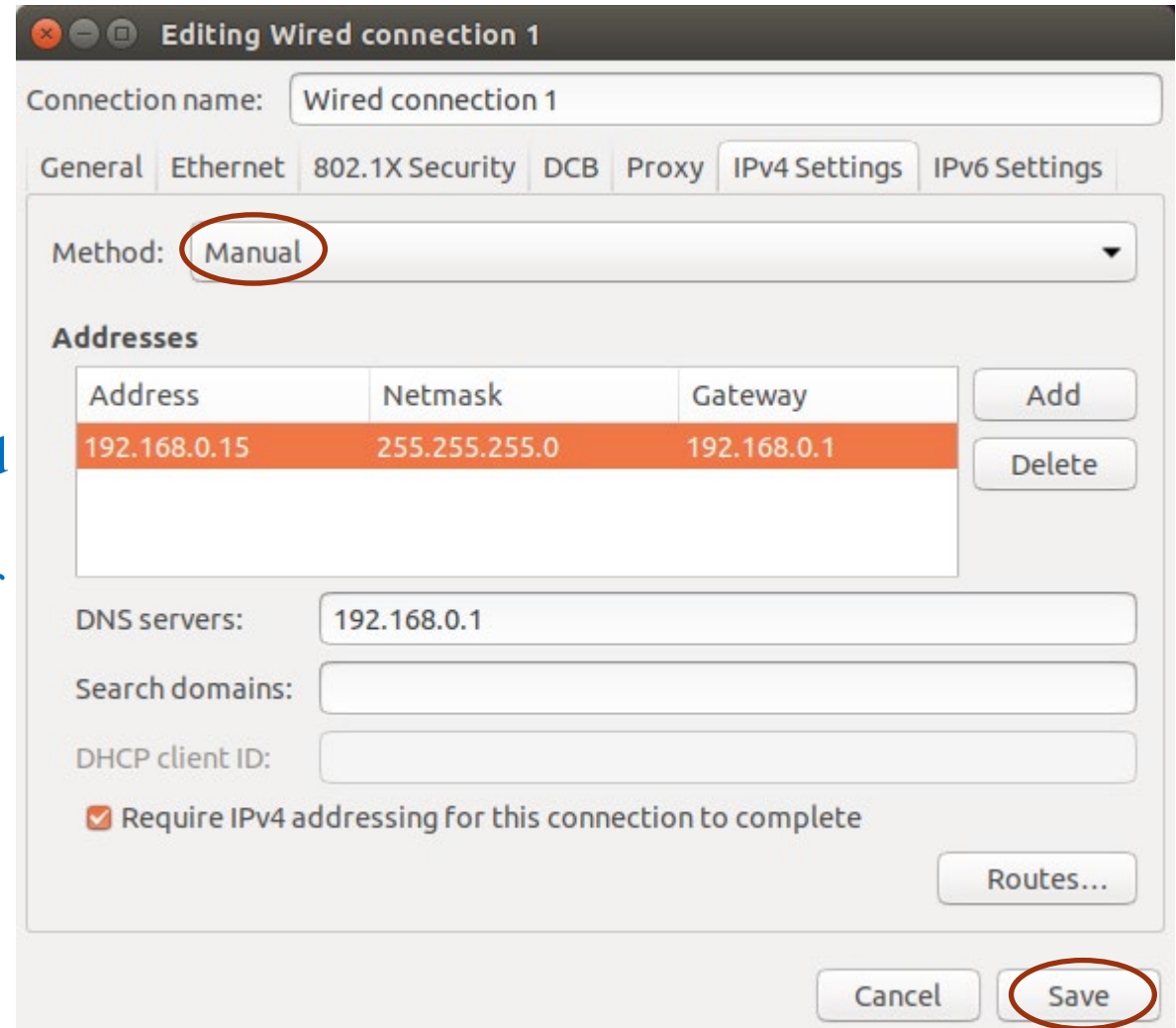
Size: 60xD60xH87 mm,
Light Weight: 370 g
Detection range: 30 m
Measurement steps: 1440



4.1 Hokuyo UST-10LX + Jetson

❑ Setup Ethernet connection with Jetson

- ❖ Make sure Ethernet cable is connected on Orbitty board and power is on;
- ❖ Login to Jetson device, and go to internet connection icon on the top right corner;
- ❖ Click on Edit Connection and select wired connection 1, in the IPv4 tab, select **Manual** and add the IP address as shown in figure.
- ❖ This is because the factory default IP address of the Lidar is 192.168.0.10 and for the Jetson or the host computer is 192.168.0.15, both are in subnet 0. Save the settings and make sure the wired connection 1 is established.
- ❖ In a terminal, run `$ping 192.168.0.10` Shall see transmitted packets echo back to the terminal.



Editing Wired connection 1

Connection name: Wired connection 1

General Ethernet 802.1X Security DCB Proxy IPv4 Settings IPv6 Settings

Method: **Manual**

Addresses

| Address | Netmask | Gateway |
|--------------|---------------|-------------|
| 192.168.0.15 | 255.255.255.0 | 192.168.0.1 |

DNS servers: 192.168.0.1

Search domains:

DHCP client ID:

☒ Require IPv4 addressing for this connection to complete

Routes...

Cancel **Save**

Alternatively, Linux guru may follow the instructions here: <https://www.jetsonhacks.com/2018/02/21/racecar-j-hokuyo-ust-10lx-configuration/>



4.1. Hokuyo UST-10LX + Jetson

- ❑ Configure Lidar Under ROS (modified from <http://fltenth.org/code.html#s1t5> (part 1)

In a terminal run

`$sudo apt-get install ros-%DISTro%-urg-node` (where you replace %DISTro% by “melodic”)

- ❑ View the data with Rviz:

Terminal 1: `$roscore`

Terminal 2: `$roslaunch urg_node urg_node _ip_address:="192.168.0.10"` (thanks to the link in notes)

Terminal 3: `$rostopic echo /scan`

Terminal 4: `$roslaunch rviz rviz -d `rospack find urg_node`/hokuyo_test.vcg` (blue part is optional)

- When rviz opens, click the “Add” button at the lower left corner. A dialog will pop up; from here, click the “By topic” tab, highlight the “LaserScan” topic, and click OK.
- Under “Global Settings”, in the “Fixed Frame” window, enter “laser”; (thanks to Andrew who figured this out)
- rviz will now show a collection of points (a point cloud) of the LIDAR data in the gray grid in the center of the screen. The points appear in colors ranging from green to red, with green points being closest to the LIDAR and red points being farthest away.
- Try moving a flat object, such as a book, in front of the LIDAR and to its sides. You should see a corresponding flat line of points on the rviz grid. Try picking the car up and moving it around, and note how the LIDAR scan data changes. Save the data or figures if needed.

4.2 Hokuyo UTM-30XL + Jetson (optional)

❑ Connect the UTM-30XL to Orbitty USB hub and power on

❑ Flow instructions here

http://wiki.ros.org/hokuyo_node/Tutorials/UsingTheHokuyoNode

- Replace %DISTro% by “melodic”.
- Replace hokuyo in all commands by urg.
- Summary of commands:

Terminal 1: `$sudo apt-get install ros-melodic-urg-node`

`$ls -l /dev/ttyACM*`

`$sudo chmod a+rw /dev/ttyACM0`

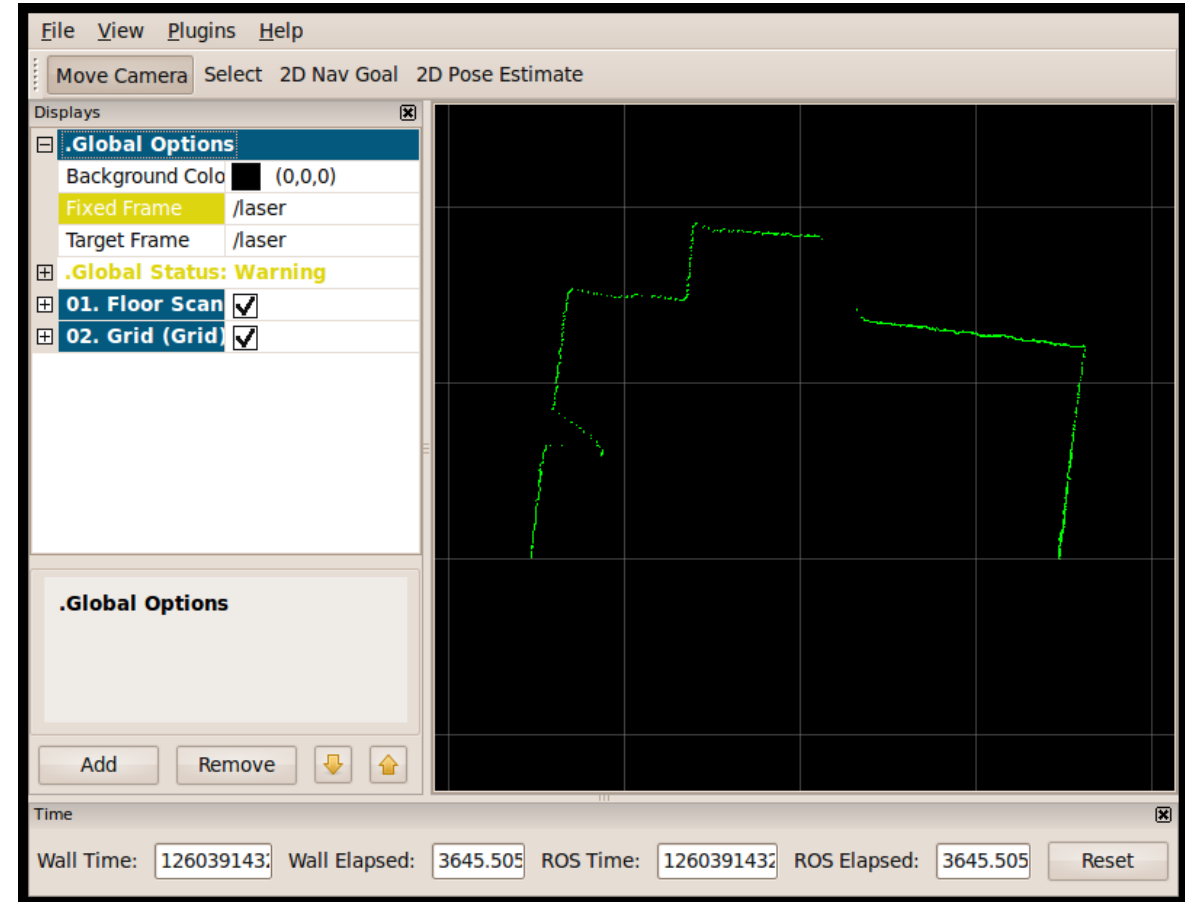
Terminal 2: `$roscore`

Terminal 3: `$roslaunch urg_node urg_node`

Terminal 4: `$rostopic echo /scan`

Terminal 5: `$roslaunch rviz rviz -d `rospack find urg_node` /hokuyo_test.vcg`

The blue part is optional. The rest is the same as UST-10XL.



Rviz example



Lab 1 Assignment: ROS Simulation

❑ In this course, we will use **Python 2**. Make sure you have installed python

1. Learn through the two ROS tutorials:

- ❖ Create a catkin workspace: http://wiki.ros.org/catkin/Tutorials/create_a_workspace and name your workspace <team#_ws> instead of <catkin_ws>, note what folders and files are created by catkin_make.
- ❖ Create a package: <http://wiki.ros.org/ROS/Tutorials/CreatingPackage> and name your package <team#_roslab1> instead of <beginner_tutorials>, note what folders and files are created by catkin_create_pkg.

2. Implement a publisher and a subscriber (python)

- ❖ 2.0 run the tutorial: <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>
- ❖ 2.1 Run racecar_simulator: download the racecar_simulator to your workspace, follow instructions in Lab1README.md. Possible errors:

Error message: “CMake Error at /opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig.cmake:113 (message):
Project 'cv_bridge' specifies '/usr/include/opencv' as an include dir, which is not found.”

Solution: edit the file to change the opencv installation dir: \$sudo vim /opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig.cmake
On line 96: set(_include_dirs "include;/usr/include;/usr/include/opencv4"); add the number “4” there, then save and exit.

For vim tutorials: https://www.radford.edu/~mhtay/CPSC120/VIM_Editor_Commands.htm

- ❖ 2.2 Subscribe to Lidar data: Your task is to create a new node which subscribes to the /scan topic and print out the nearest= min(ranges) and farthest=max(ranges) data values.

Lab 1 Assignment: ROS Simulation

3. Implement custom messages

- ❖ Refer to “create custom message files” tutorial: <http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv>

4. Record and publish bag files

- ❖ Rosbag tutorial: <http://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data>

5. Use launch files to launch multiple nodes

- ❖ Rviz tutorial: <http://wiki.ros.org/rviz/Tutorials>
- ❖ Roslaunch tutorial: <http://wiki.ros.org/roslaunch> set your launch file as team#_lab1lidar.launch

❑ Pay attention to programming styles

- ❖ Refer to PDB tutorial: <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>
- ❖ Python-ROS style guide: <http://wiki.ros.org/PyStyleGuide>:
 1. Use class objects as shown in the skeleton codes provided. You may add new functionality inside the skeleton class.
 2. Keep variables private as much as you can. Global variables are strongly discouraged.
 3. Use spaces instead of tab to indent. Spaces are universally the same in all machines and text editors, but tabs are not.
 4. Vectorize your code to avoid nested loops, as for-loops slow down your code. Numpy is extremely helpful in vectorizing loops.