RECORDING SYSTEM HANDBOOK

Daniel Zukerman

Sahar Pardo

Sensor System Setup - Table of Contents

1. Initial Explanations and Setup Explanation:

Introduction:

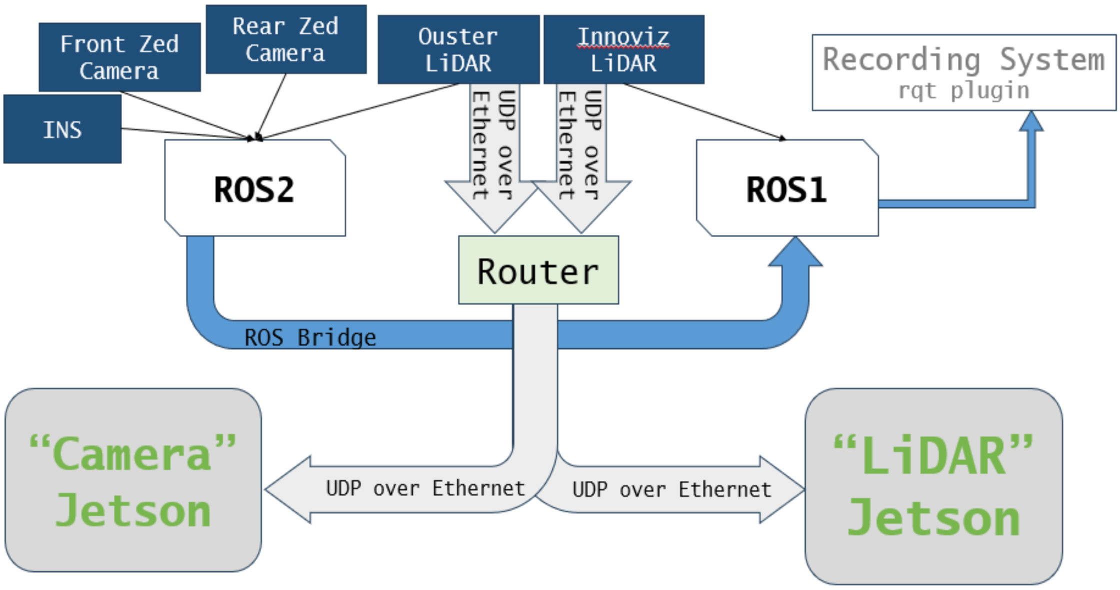
We strongly suggest reading and making sure you understand what is going on in this project regarding the operation and relation of ROS2 and ROS1, and how to operate linux.

We created a GUI that centralizes visualization of the sensors – it is a plugin in rqt in ROS1. The reason for keeping it in ROS1 is that the experimental Rviz plugin for rqt is only available on ROS1, and that we did not have enough time to migrate the GUI plugin to ROS2 – it may take some time.

* As of 31.5.25 ROS Noetic has reached its EOL – which leads us to suggest that the next project will from migrating the GUI to ROS2.

Setup Explanation:

(explain which sensors are connected through which ROS to which computer)



2. ROS1 Noetic Installation:

(source: noetic/Installation/Ubuntu - ROS Wiki)

Setup your computer to accept software from packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
```

Set up your keys

```
sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo
apt-key add -
```

Installation:

First, make sure your Debian package index is up-to-date:

```
sudo apt update
sudo apt install ros-noetic-desktop-full
```

Environment setup: You must source this script in every bash terminal you use ROS1 in.

```
source /opt/ros/noetic/setup.bash
```

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator
python3-wstool build-essential
```

Before you can use many ROS tools, you will need to initialize ROSdep. ROSdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS. If you have not yet installed rosdep, do so as follows.

```
sudo apt install python3-rosdep
```

Initialize rosdep

```
sudo rosdep init
rosdep update
```

3. ROS2 Foxy installation:

(source: GitHub - shodlly/ROS-2-Foxy-Installation-Guide)

4. ROS Bridge installation:

If you added any ROS sourcing lines to your bashrc file, comment them out first. Then, in a new terminal (so the bashrc changes will be applied) install ROS1 Bridge:

```
sudo apt install ros-foxy-ros1-bridge
```

To use the bridge:

1. Run roscore in a terminal with ROS1 sourced (roscore is run automatically upon launching a launch file, but can be also run manually)
2. In another terminal, source ROS1 and then ROS2
3. Run the following command in the terminal with both distros sourced:

```
ros2 run ros1_bridge dynamic_bridge --bridge-all-topics
```

4. Now ROS1 nodes should be seen in ROS2 and vice versa. To check that:
   a. Open two terminals and source ROS1 in them. In one of them, run:

   ```
   roscore
   ```

   b. Open another terminal and source both ROS distros (like in step 2), then run the bridge command from step 3
   c. Open a fourth terminal, and source ROS2. Then run in the ROS1 terminal where roscore is not running:

   ```
   rosrun rospy_tutorials talker
   ```

   d. In the terminal with ROS2 source, run "ros2 topic list". You should see the chatter topic. Also, you can run:

   ```
   ros2 run demo_nodes_cpp listener
   ```

   And see if you get any reaction. You should get what's sent on the "chatter" topic.

5. Ouster LiDAR

https://github.com/ouster-lidar/ouster-ros/tree/ros2-foxy

Requirements: we worked on ROS2 foxy, below is how to download some required packages:

```
sudo apt install -y

  ros-foxy-pcl-ros

  ros-foxy-tf2-eigen

  ros-foxy-rviz2


sudo apt install -y       \

  build-essential       \

  libeigen3-dev         \

  libjsoncpp-dev        \

  libspdlog-dev         \

  libcurl4-openssl-dev   \

  cmake                 \
```

building the ouster package:

```
mkdir -p ros2_ws/src && cd ros2_ws/src

git clone -b ros2-foxy --recurse-submodules https://github.com/ouster-lidar/ouster-ros.git

source /opt/ros/foxy/setup.bash

cd ros2_ws

colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release

source ros2_ws/install/setup.bash
```

Launching the Ouster and Visualization:

```
cd ros2_ws
```

When connected directly to the Jetson:

```
ros2 launch ouster_ros record.launch.xml  sensor_hostname:="os-122214001580.local"
lidar_mode:=1024x10
```

When connected to the router:

```
ros2 launch ouster_ros record.launch.xml  sensor_hostname:=192.168.1.201
lidar_mode:=1024x10
```

Visualization:

```
ros2 run rviz2 rviz2
```

- set fixed_frame to os_lidar
- add pointcloud2, topic: /ouster/points

6. ZED Camera

SDK installation:

in this link https://www.stereolabs.com/docs/installation/jetson you can download the Zed cameras SDK.

Cameras PU and Visualization:

```
cd ros2_ws/src/zed-ros2-wrapper/

ros2 launch zed_wrapper dual_zed_launch.py

ros2 run rviz2 rviz2
```

- ➔ set fixed frame to:zed_front_camera_link (front/rear)
- ➔ topic: /zed_rear/zed_node/rgb/image_rect_color (front/rear)

front camera serial number: 39453324
rear camera serial number: 35014157

7. INS

https://us.inertiallabs.com:31443/projects/INS/repos/inertiallabs-ros2-pkgs/browse

First we will need to build the package for the INS:
Requirements:
we will download the required addons for the INS, and will built it:

```
cd  ros2_ws/src

git clone https://us.inertiallabs.com:31443/scm/ins/inertiallabs-ros2-pkgs.git

cd ros2_ws

colcon build
```

INS Power up:
the following line will get the INS to run (if you do it for the first time do the commands bellow this one and then try it)and the topic will be aired:

```
cd ros2_ws/
ros2 run inertiallabs_ins il_ins --ros-args -p ins_url:=serial:/dev/ttyUSB0:115200 -p
ins_output_format:=102
```

# the output format can be changed to 51, we preferred to work with 102.
if the INS is not up, run these commands and then go back to the ros2 run command:

```
sudo chmod 666 /dev/ttyUSB0

sudo stty -F /dev/ttyUSB0 115200

source install/setup.bash
```

INS visualization:

Make sure that the topic is on (can be check by 'ros2 topic list'), and if so open another terminal and run this command:

```
ros2 topic echo /Inertial_Labs/sensor_data
```

You should see now the data running in the new terminal you opened.

8. Innoviz LiDAR

API download:

zuki

Important: this sensor is unlike all the others is running on ROS1 and not ROS2. So the sensor PU will be with ROS1- make sure you got noetic.

Sensor PU:

open terminal and run the next commands:

```
source /opt/ros/noetic/setup.bash

roscore
```

open new terminal (make sure roscore is running in the background) and run the following commands:

```
cd ~/catkin_ws/src/innoviz/innovizapi_5.6.2_0857bc7c2/ros/catkin_ws

roslaunch ./src/launch/innoviz_ros.launch
```

Sensor Visualization:

Make sure the the topics relevant to the innoviz are up (using rostopic list) and in a third terminal run the following:

```
ros2 run rviz2 rviz2
```

➔ set fixed frame to:base_link

➔ topic: /invz_reflection_0

- GUI page
- Finish the innoviz (API)
- Foxy

- Current versions
- Ips and network