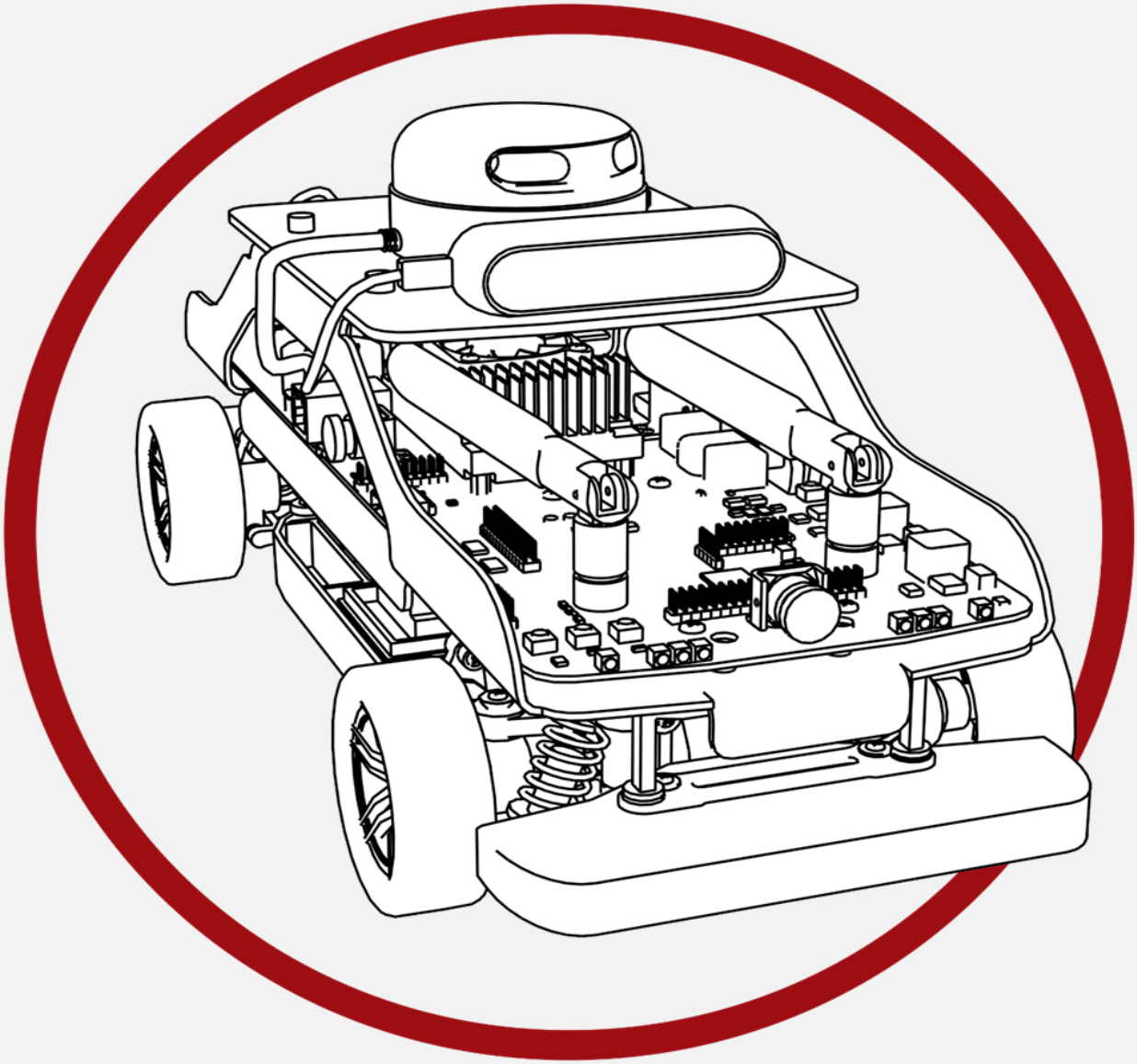
QUANSER
INNOVATE·EDUCATE



Self-Driving Car Studio

Software – ROS User Manual

v 1.0 — 9*th Jan* 2023

For more information on the solutions Quanser Inc. offers, please visit the web site at: http://www.quanser.com

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6, Canada

info@quanser.com
Phone : 19059403575
Fax : 19059403576
printed in Markham, Ontario.

FCC Notice This device complies with Part 15 of the FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Industry Canada Notice This Class A digital apparatus complies with Canadian ICES-003. Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

Waste Electrical and Electronic Equipment (WEEE)

This symbol indicates that waste products must be disposed of separately from municipal household waste, according to Directive 2002/96/EC of the European Parliament and the Council on waste electrical and electronic equipment (WEEE). All products at the end of their life cycle must be sent to a WEEE collection and recycling center. Proper WEEE disposal reduces the environmental impact and the risk to human health due to potentially hazardous substances used in such equipment. Your cooperation in proper WEEE disposal will contribute to the effective usage of natural resources.

CE Compliance CE

This product meets the essential requirements of applicable European Directives as follows:

- 2006/95/EC; Low-Voltage Directive (safety)
- 2004/108/EC; Electromagnetic Compatibility Directive (EMC)

Warning: This is a Class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

Caution

This equipment is designed to be used for educational and research purposes and is not intended for use by the public. The user is responsible to ensure that the equipment will be used by technically qualified personnel only. While the end-effector board provides connections for external user devices, users are responsible for certifying any modifications or additions they make to the default configuration.

Table of Contents

A. Overview

Prior to running sample ROS applications please read through the User Manual - Software Python before continuing with this document. The ROS code examples and this user manual are using Python and our Quanser Python Library. C++ functions are available for **advanced users**. By default the QCar comes with ROS (melodic) and ROS(Dashing) installed.

Note: The ~/.bashrc does not auto-source either of the two ROS distributions as this is application specific. If you wish to stick to one development environment change the ~/.bashrc to source the specific ROS distribution you'll use.

Note: This user manual assumes users have the fundamental knowledge of ROS.

Please check https://docs.ros.org/en/dashing/index.html for help with ROS Dashing, or

The overall Node design process is described in Figure 1 below.

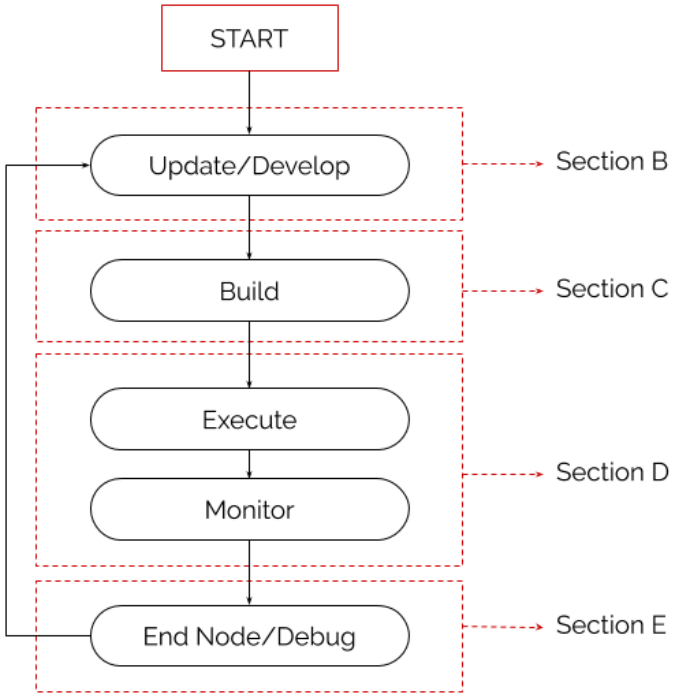


Figure 1. Process diagram for ROS code deployment

B. Update/Development Details

Please refer to the **Quick Start Guide** for links regarding the latest ROS software packages for the QCar.

Please check **Section B** in User Manual - Software Python to make sure all the python modules and Quanser Application Libraries are installed. When developing ROS1 or ROS2 nodes, please use the **python3** environment for better usage of our python library.

```
#!/usr/env python3
```

ROS nodes can be found in the following location **examples\self_driving_car_studio\qcar**

For ROS Melodic:

To check the list of installed ROS1 packages please run the following command:

```
nvidia@qcar-*****:~$ rospack list-names
```

For ROS Melodic look for the subfolders called:

- ros1_cpp
- ros1_python

Copy the content of either folder to the ros1 directory on the QCar. Keep in mind you cannot run both the python and C++ nodes from the same ROS1 workspace on the QCar.

For ROS Dashing:

To check the list of installed ROS2 packages please run the following command:

```
nvidia@qcar-*****:~$ ros2 pkg list
```

For ROS Dashing look for the subfolder called:

- ros2

Copy the content of the ros2 folder to the ros2 directory on the QCar.

C. Build

For ROS1

When building the package for the first time, please make sure **catkin_make** uses python3:

```
nvidia@qcar-****:~$ catkin_make PYTHON_EXECUTABLE=/usr/bin/python3
```

Later builds can run without specifying your python version as it has set python3 as default python environment. If you only want to build one single ROS package you can add **--pkg package_name** after **catkin_make**. After finishing building, please make sure to source your devel/setup.bash file.

NOTE: Do not run catkin_make while in sudo authority. The best solution we have is for you to make a new ros1 workspace.

For ROS2

When building the ros2 package for the first time, please use the **colcon build** command:

```
nvidia@qcar-****:~$ colcon build --symlink-install
```

NOTE: Do not run colcon build while in sudo authority. The best solution we have is for you to make a new ros2 workspace.

D. Executing and Monitoring Nodes

For ROS1

Nodes can be ran using a **.launch** file or individually.

Sequence of steps for running nodes individually:

Terminal session 1

1. Open a terminal session and source ros using the command:

```
nvidia@qcar-*****:~$ source /opt/ros/melodic/setup.bash
```

2. Start the roscore server

```
nvidia@qcar-*****:~$ roscore
```

Terminal session 2 (or more)

1. Switch the terminal to sudo authority:

```
nvidia@qcar-*****:~$ sudo -s
```

2. Source the ROS

```
nvidia@qcar-*****:~# source /opt/ros/melodic/setup.bash
```

3. Use rosrun to run the specific node

```
nvidia@qcar-*****:~# rosrun qcar <qcar node name>
```

if you are viewing your ROS nodes from a Ground Control Station, please make sure to have **XLaunch** set up properly. See the User Manual – Connectivity for more information on this.

Use standard command **rostopic list/echo/hz/info** to monitor the topics that each node is sending/receiving.

Hint: **Use image_view to view camera streams**

For ROS2

Terminal session 1 (or more)

1. Switch the terminal to sudo authority:

```
nvidia@qcar-*****:~$ sudo -s
```

2. Source ROS

```
nvidia@qcar-*****:~# source /opt/ros/dashing/setup.bash
```

3. Run each ROS node

```
nvidia@qcar-*****:~# ros2 run qcar <node name>
```

E. Stop Node & Troubleshooting

Press **Ctrl + C** to stop any ROS node/launch files. If the terminal hangs after pressing **Ctrl + C**, please press **Ctrl + Z** to refresh the terminal. When stopping a ROS node that involves motor and/or Lidar, please run the python script with **sudo** authority called **HardwareStop.py.** In the provided resources this script can be found under **examples/self_driving_car_studio/qcar/hardware_tests**

For general troubleshooting, users can still use the **try/except/finally** structure in their ROS codes to get error messages. Please view **Section E** in User Manual - Software Python for more information on this structure.

For ROS troubleshooting, users can use either the go-to command **roswtf** or look for more information at http://wiki.ros.org/ROS/Troubleshooting.

F. Ros-to-Ros Communication

The QCar was built around ROS2, but with the vast libraries built around ROS1, sometimes you need to mix ROS1 and ROS2 nodes. ROS-to-ROS communication can be done in multiple ways. This section will explain how to setup a QCar-to-Ubuntu PC communication using a ROS2 package called **ros1_bridge**.

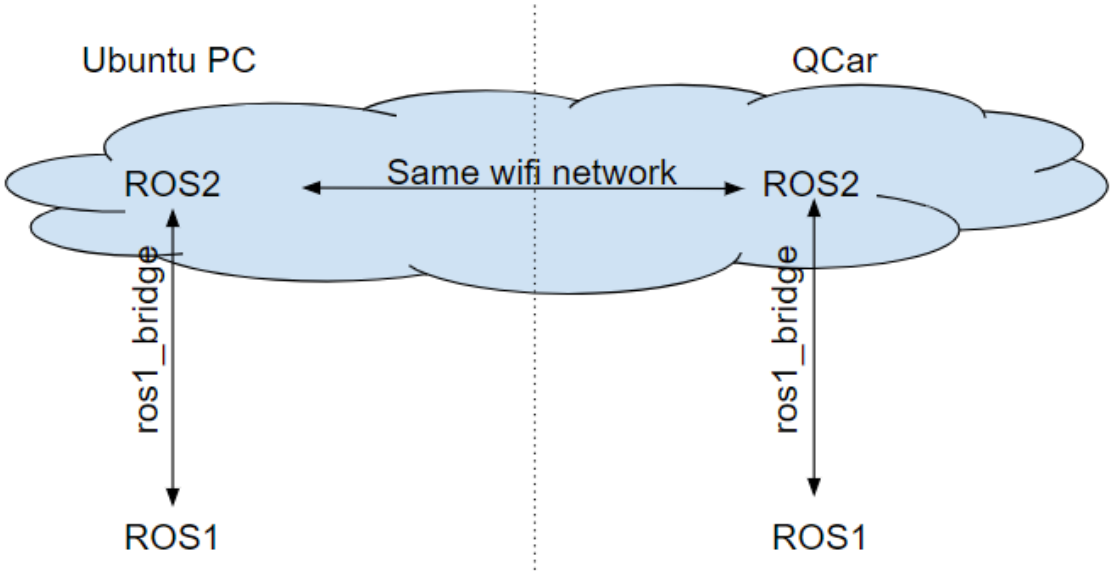


Figure 1: Visual Diagram of QCar-to-Ubuntu PC communication using a ros1_bridge

All the topics that are published by a ROS2 device can be subscribed by other ROS2 devices if they are in the same network.

Only Using ROS2?

If you are just using ROS2 as your main development platform and are running ROS2 on both QCar and the Ubuntu PC, then the ROS-to-ROS communication does **not need extra setup**.

Using ROS1 on either or both platform?

If users will also use ROS1 on either or both platforms, then they need to download a ROS2 package called **ros1_bridge.** This package will bring all the topics from ROS1 environment to ROS2 or vice-versa.

A typical use case is a ROS1 node on the QCar capturing a CSI image which needs to be used by a ROS 1 node on a remote machine. Using the **ros1_bridge** on the **QCar,** ROS2 on the **QCar** will bring that topic from ROS1 and share it with the other ROS2 device **Ubuntu PC** on the same network. The **ros1_bridge** on **Ubuntu PC** will detect that topic in the *ros2 topic list* and bring it to ROS1 on the **Ubuntu PC** where it can process the image for its application like reading a sign, lane detection, etc. The resulting command will then use the same tunnel to return to the ROS1 environment on QCar**.**

When measuring the performance, WiFi latency should be considered. See User Manual – Connectivity for discussion on WiFi configurations.

Please follow this link https://github.com/ros2/ros1_bridge/blob/master/README.md to setup **ros1_bridge** on each device.

Solutions for teaching and research. Made in Canada.