Mercator–RVR: SLAM Map Fusion

Salman Houdaibi, based on the work of Edwige Loems

June 2025

Abstract

This document is a hands-on, step-by-step guide explaining how to deploy, update and execute the `rvr_ros`-based software stack that accompanies the Mercator platform—an enhanced Sphero RVR designed for swarm SLAM research. The target audience is anyone who already owns at least one Mercator robot (Raspberry Pi 4 mounted on a Sphero RVR, plus YDLIDAR X4, Terabee Multiflex proximity sensors and an OAK-D camera) and wants to reproduce the experimental pipeline described in *Mercator: hardware and software architecture for experiments in swarm SLAM* (Kegeleirs *et al.*, IRIDIA Technical Report TR/IRIDIA/2022-012).

Contents

1 Hardware Assumptions

- Sphero RVR with factory firmware $\geq$ v4.1.
- Raspberry Pi 4 Model B (2 GB or 4 GB) powered through the official RVR UART ribbon cable.
- Sensors: YDLIDAR X4 (USB), Terabee Multiflex 8 (USB), optional Luxonis OAK-D (USB 3.0).
- On-board Wi-Fi *and* wired Ethernet exposed via the RVR's USB-to-Ethernet dongle for ROS multi-master.

Battery policy **Never run the stack with fewer than three battery bars on *either* the RVR main battery *or* the OAK-D's companion pack.** Low voltage can silently disable the IMU, set odometry velocities to 0, or brown-out the depth camera.

2 Base System : Ubuntu 20.04 LTS

Flash & first boot

1. Flash **64-bit** Ubuntu 20.04 Server for `rpi_4`.
2. Add `ssh` and an `ubuntu:ubuntu` user with sudo privileges.
3. Boot on the RVR, resize filesystem, `sudo apt update && sudo apt full-upgrade`.

3 Install ROS Noetic

See the Technical Report for more information

```
sudo apt install ros-noetic-tf2-ros ros-noetic-tf2-tools \
                 ros-noetic-gmapping ros-noetic-map-server \
                 ros-noetic-diagnostic-updater \
                 python3-serial python3-opencv python3-scipy python3-skimage
```

4 Update the Sphero SDK

Mandatory to use the drivers correctly.

```
git clone https://github.com/sphero-inc/sphero-sdk-raspberrypi-python ~/sphero-sd
cd ~/sphero-sdk
pip3 install --user -r requirements.txt
python3 setup.py install --user
```

5 `rvr_ros` Workspace

Create and populate catkin_ws

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone https://github.com/rafftod/rvr_ros
git clone https://github.com/rafftod/demiurge-rvr-dao -b ros
git clone https://github.com/rafftod/ARGoS3-AutoMoDe -b ros
rosdep install --from-paths . --ignore-src -r -y
```

Drop-in your custom Python nodes

Copy every `*.py` from the repository to :

`~/catkin_ws/src/rvr_ros/src/`

For instance:

```
cp ~/Downloads/random_walk.py          ~/catkin_ws/src/rvr_ros/src/
cp ~/Downloads/odom_imu_corrector.py   ~/catkin_ws/src/rvr_ros/src/
cp ~/Downloads/map_sharing_node.py     ~/catkin_ws/src/rvr_ros/src/
```

6 Keeping ROS Noetic and the SDK fresh

```
cd ~/sphero-sdk       && git pull && python3 setup.py install --user
cd ~/catkin_ws/src    && git pull --recurse-submodules
rosdep install --from-paths . --ignore-src -r -y
cd ~/catkin_ws        && catkin_make
```

7 Network Configuration for Multi-Master

Static Ethernet link

Master Pi	Slave Pi(s)
10.66.66.1/24	10.66.66.2/24

```
# Master
sudo ip link set eth0 up && sudo ip addr flush dev eth0
sudo ip addr add 10.66.66.1/24 dev eth0
export ROS_MASTER_URI=http://10.66.66.1:11311
export ROS_IP=10.66.66.1

# Slave   (change .2 -> .3, .4      as needed)
sudo ip link set eth0 up && sudo ip addr flush dev eth0
sudo ip addr add 10.66.66.2/24 dev eth0
export ROS_MASTER_URI=http://10.66.66.1:11311
export ROS_IP=10.66.66.2
```

Append the last two `export` lines to `~/.bashrc` or gather them in `ros_net.sh` and source it from `~/.bashrc`.

8 Runtime Recipe

All commands below assume `roscore` is already running on the master and the relevant `setup.bash` has been sourced in every terminal.

8.1 Low-level drivers

1. **Sphero RVR driver**
   `rosrun rvr_ros rvr_async_driver.py`
2. **YDLIDAR X4**
   `roslaunch ydlidar_ros_driver X4.launch scan:=/rvr1/scan`
3. **Terabee proximity array**
   `rosrun teraranger_array teraranger_multiflex _portname:=/dev/ttyACM0`
4. **IMU–Odometry aligner (optional but recommended)**
   `rosrun rvr_ros odom_imu_corrector.py`

8.2 High-level behaviour

- `roslaunch rvr_ros gmapping.launch robot_ns:=rvr1`
  creates TF chains, launches SLAM GMapping, and continuously saves a map under `~/maps/`.
- `roslaunch rvr_ros random_walk.launch robot_name:=rvr1`
- `roslaunch mon_package map_sharing.launch robot_id:=rvr1`

Be Careful Each node set belongs in its own terminal tab or `tmux` pane. Mixing drivers and SLAM in the same shell hides runtime errors.

9 Troubleshooting Checklist

1. **No odometry**: battery below three bars ⇒ replace or recharge.
2. **Frozen TF**: confirm that `/tf_static` contains two static publishers (base→footprint, footprint→laser).
3. **Bad SLAM drift**: ensure `odom_imu_corrector.py` is running *before* `slam_gmapping`.
4. **Lost ROS topics across robots**: wrong `ROS_IP` or Netmask; check with `ifconfig` and `echo $ROS_MASTER_URI`.

10 References

1. M. Kegeleirs, R. Todesco, D. Garzon Ramos, G. Legarda Herranz, and M. Birattari, *Mercator: hardware and software architecture for experiments in swarm SLAM*, IRIDIA Technical Report TR/IRIDIA/2022-012, Nov. 2022.
2. Official ROS Noetic documentation: `https://wiki.ros.org/noetic`
3. Sphero SDK for Raspberry Pi: `https://github.com/sphero-inc/sphero-sdk-raspberrypi-python`