MYBOTSHOP

ROBOTICS

ROVO2

Tracked Robot — User Manual

CONTACT

info@mybotshop.de

support@mybotshop.de

mybotshop.de

Contents

1 Important Notice

Rovo Barkhausen

ROVO2

Warning: Please thoroughly review this document. If uncertain, it's crucial to seek guidance from specialists, experts, or the manufacturers of the components used. The robot should not be activated until clarification is obtained. If uncertain, please refer to the provided guides or reach out to a specialist, expert, or manufacturer associated with the components used. The robot should not be activated until uncertainties are resolved. Contact details can be found in the support forums within the Getting Help Section. Only individuals familiar with robotics or these instructions should operate the robot. Untrained users may pose risks when handling robots.

Important: The robot supplied by MYBOTSHOP GmbH serves as an R&D device and does not possess CE Marking or Certificate of Incorporation. Familiarity with basic ROS concepts is necessary. For those unfamiliar with ROS, we advise referring to the ROS Wiki as an initial step. Kindly note that the aforementioned robot is categorized as a partially completed machine according to the Machinery Directive 2006/42/EC and does not bear a CE marking. Clients understand and accept that any information or materials furnished by MYBOTSHOP GmbH are exclusively for R&D purposes. Services are provided on an "AS IS" basis without any express or implied representation or warranty, including warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranty. MYBOTSHOP GmbH is not liable for any damages, be they direct, indirect, special, incidental, or consequential, arising from the use or inability to use the provided information or materials. This limitation of liability applies irrespective of the nature of the action, whether contractual, tortious, or otherwise.

2 Disclaimer

The ROVO2 documented here is sold and supported by *MYBOTSHOP GmbH*, built on the **HAWE Mattro ROVO** tracked-vehicle platform with MYBOTSHOP's own ROS 2 software layer on top. Basic ROS 2 understanding is required to operate the software stack covered in this manual — if you are not familiar with ROS 2, we recommend checking the ROS 2 documentation first.

The client acknowledges and agrees that any information or materials provided by MYBOTSHOP GmbH are for R&D and development purposes only. Any services are provided "AS IS" and without any representation or warranty of any kind, express or implied, including but not limited to any warranty of merchantability, fitness for a particular purpose, non-infringement, or any other warranty.

This manual documents both the current ROS 2 Jazzy software stack (v2.0, December 2025) and a legacy ROS Noetic / ROS 2 Foxy dual-stack quick-start kept as a supplementary appendix for earlier-generation and customer-specific units — always confirm which stack is installed on your unit before following a command from the wrong chapter.

MYBOTSHOP GmbH shall not be liable for any damages, including but not limited to direct, indirect, special, incidental, or consequential damages, arising out of or in connection with the use or inability to use the information or materials in this document. This limitation on liability shall apply regardless of the form of action, whether in contract, tort, or otherwise.

3 About the ROVO2

ROVO2 Tracked Robot

ROVO2

The MBS ROVO2 is an electric motor driven tracked vehicle developed for use in unpaved terrain. The manufacturer of the drive is a renowned company trusted by many companies and organizations. The drive is designed for high sub-zero temperatures, making it popular with mountain rescue and expedition vehicles. The entire vehicle is protected to IP65.

Key Features:

- Dual 7.5KW electric motors with 1:16 gear reduction
- Top speed of 20km/h
- Battery range of approximately 8 hours or 40km
- Low center of gravity due to integrated chassis battery
- ROS2 Jazzy support with Gazebo Harmonic simulation
- Web-based control interface

Current Software Version: v2.0 (December 2025)

Important: These tutorials assume that you are comfortable working with ROS. We recommend starting with the ROS tutorial if you are not familiar with ROS already.

4 Quick Start – Network, Webserver & Power

Pre-requisites

Before operating the ROVO2, ensure you have:

- Read and understood the safety guidelines
- Access to the robot's network (wired or wireless)
- A computer with ROS2 Jazzy installed (for visualization)

Robot App

The ROVO2 comes with pre-installed ROS2 software. The robot's onboard computer runs Ubuntu 24.04 LTS with ROS2 Jazzy. Core drivers and services start automatically on boot via systemd services.

Robot Webserver

The ROVO2 features a web-based control interface accessible at:

URL: http://192.168.131.1:9000

Default Credentials:

- Username: admin
- Password: mybotshop

The webserver provides:

- Real-time robot visualization (3D model)
- Joystick teleoperation
- Service management (start/stop ROS nodes)
- Map visualization
- Battery monitoring
- Rosbag recording
- VNC remote desktop access

Charging Robot

To charge the ROVO2:

1. Power off the robot
2. Connect the charger to the charging port
3. Monitor the charging status via the charger LED
4. Disconnect when fully charged

Emergency Stop

The ROVO2 is equipped with emergency stop buttons. When pressed:

- All motor power is immediately cut
- The robot enters a safe parking state
- Services continue running but motion commands are ignored

To resume operation after emergency stop:

1. Clear the emergency condition
2. Release the emergency stop button
3. Reset the gear to neutral, then re-engage

Power On

To power on the ROVO2:

1. Ensure the emergency stop is released
2. Press the main power switch
3. Wait for the system to boot (approximately 30-60 seconds)
4. Verify status via the webserver or SSH

Power Off

To safely power off the ROVO2:

1. Set the gear to neutral (gear 0)
2. Stop all running applications
3. Press the main power switch to turn off

Basic Operation

Remote Control

The ROVO2 supports multiple teleoperation methods:

1. ROVO Controller - Dedicated hardware controller
2. Logitech Joystick - Requires deadman switch
3. Web Interface - Browser-based joystick control
4. Keyboard Teleop - ROS2 teleop_twist_keyboard

Network Interface

Network Table

IP Address	Device	Username	Password
192.168.131.1	Robot MCU	robot	mybotshop
192.168.131.1:9000	Webserver	admin	mybotshop
192.168.131.150	Steamdeck	deck	mybotshop
192.168.131.200	Router	SSID	mybotshop
192.168.131.200	Router-Web	admin	Admin1232025
192.168.131.20	Ouster Lidar	x	x

Warning: Do not set your computer's IP to any of the above reserved addresses.

Note: The robot login credentials are: Username: robot Password: mybotshop

Ethernet Connection

1. Power on the robot
2. Connect a LAN cable to the robot's ethernet port (topmost port on demo units)
3. Configure your computer's network settings

_images/rovo_net.webp

ROVO2 Ethernet Connection Port

Static Network Connection

For first-time connection, configure a static IP on your computer:

1. Go to Settings > Network > + (add new connection)
2. Select Manual in IPv4 settings
3. Set the following: Address: 192.168.131.51 Netmask: 24
4. Save and restart your network

Verify the connection:

```
# Check your local IP
ifconfig

# Ping the robot
ping 192.168.131.1

# SSH into the robot
ssh -X robot@192.168.131.1
Password: mybotshop
```

5 ROS 2 Software – Setup & Operation

Pre-requisites

- Safety Guidelines Autonomous Mobile Robot .
- Network Interface Pre-requisites .

Installation

Pre-requisite Software

The ROS driver for the robots is initially supported for the ROS Noetic . All of the ROS-related software should be installed either on remote PC/Nvidia board on the robot. The steps shown below are for remote PC both boards usually come with pre-installed ROS distribution.

Note: All the ROS related software should be installed/run on remote PC/Rasberry Pi/Nvidia board, nothing needs to be installed on the robots control board as they come pre-installed.

Ubuntu 20.04 - ROS NOETIC

_images/ros_noetic.webp

The ROS driver for the robots is initially supported for the ROS Noetic . All of ROS related software should be installed in a Remote-PC . The steps for shown below are for Remote-PC .

> **Warning:** Care should be taken when installing ROS so that it matches with your CPU's architecture (i.e. armhf, amd64, arm6, etc.)

PC Setup

ROS Noetic installation requires Ubutnu 20.04.

1. Download Ubuntu 20.04
2. Follow the Ubuntu 20.04 Installation guide

ROS-noetic Installation

The ros driver provided can be run either on a remote PC or on board robot computer. Usually, the on-board computer already has pre-installed ROS distribution, so the instructions below will be applicable to a remote computer. You may run the following commands to install ROS noetic or you can simply follow the instructions from the roswiki.

1. Enter the Ubuntu terminal and type in the following command:

```
cd
```

1. The next step is setting up the source list:

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

1. After which you will set up the keys:

```
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

1. Then, you will install ROS-noetic:

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

1. Finally, you will add ROS environment to your bash file:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

1. Install python dependencies and initialize repo:

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

Catkin Workspace

1. Ensure that you have correctly setup ROS noetic:

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

1. Install catkin_tools:

```
sudo apt install python3-catkin-tools
```

1. Create your workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin build
```

1. Enable the workspace:

```
source devel/setup.bash
```

1. Add to bashrc:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Ubuntu 18.04 - ROS MELODIC

_images/melodic.webp

The ROS driver for the robots is also supported for ROS Melodic on Ubuntu 18.04.

ROS-Melodic Installation

Run following commands to install ROS Melodic or follow the instructions .

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
sudo apt update
sudo apt install ros-melodic-desktop-full
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Installation Robot

Note: ROS2 Foxy reached end-of-life. Please consider upgrading to ROS2 Jazzy.

The ROVO2 ROS2 Foxy packages can be installed on Ubuntu 20.04.

1. Install ROS2 Foxy

Follow the official ROS2 Foxy installation guide .

2. Clone and Build

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
# Clone ROVO2 packages
cd ~/ros2_ws
colcon build --symlink-install
source install/setup.bash
```

Installation External

For external computers:

```
colcon build --symlink-install --packages-select rovo_description rovo_viz
source install/setup.bash
```

Installation Robot

The ROVO2 comes pre-installed with ROS2 Jazzy. For a fresh installation or re-installation, follow these steps:

1. Set Robot Hostname

```
sudo hostnamectl set-hostname rovo-unit-071
```

2. Create Workspace Directory

```
sudo mkdir /opt/mybotshop && sudo chown -R robot:robot /opt/mybotshop
```

3. Clone and Install

Clone the repository to the robot's PC and run the installer:

```
cd /opt/mybotshop/src/mybotshop/robot_installer
sudo chmod +x total_install.sh
sudo ./total_install.sh
```

4. Build ROS2 Workspace

```
cd /opt/mybotshop
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
source install/setup.bash
```

5. Install Startup Services

```
ros2 run rovo_autostart startup_installer.py
```

6. Disable System Suspend

```
sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target
sudo systemctl disable systemd-networkd-wait-online.service
sudo systemctl mask systemd-networkd-wait-online.service
```

7. Configure Network

Edit the netplan configuration:

```
sudo nano /etc/netplan/01-network-manager.yaml
```

Add the following configuration:

```
network:
  version: 2
  renderer: NetworkManager
  bridges:
    br0:
      interfaces:
        - eno1
        - enp3s0
      dhcp4: no
      addresses:
        - 192.168.131.1/24
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
      parameters:
        stp: false
        forward-delay: 0
      optional: true

  ethernets:
    eno1:
      dhcp4: no
      optional: true
    enp1s0:
      dhcp4: no
      optional: true
```

Apply the configuration:

```
sudo netplan apply
```

Installation External

For external (host) computers that need to visualize and interact with the ROVO2:

1. Clone the Repository

Clone the ROVO2 repository into your ROS2 workspace.

2. Build Essential Packages

```
colcon build --symlink-install --packages-select rovo_description rovo_viz
source install/setup.bash
```

3. Set Environment and Launch

```
export ROVO_NS="rovo_unit_071"
ros2 launch rovo_viz view_robot.launch.py
```

Dependencies

Install the required ROS2 packages:

```
sudo apt-get install ros-jazzy-urdf ros-jazzy-xacro \
                     ros-jazzy-joint-state-publisher ros-jazzy-robot-state-publisher \
                     ros-jazzy-joy ros-jazzy-teleop-twist-joy ros-jazzy-twist-mux \
                     ros-jazzy-navigation2 ros-jazzy-nav2-bringup \
                     ros-jazzy-ros-gz ros-jazzy-ros-gz-bridge
```

Autostart

Note: For ROS Noetic, startup is managed via roslaunch and custom scripts. See the Packages section for bringup commands.

Systemd Services

Systemd services for ROS2 Foxy follow a similar pattern to Jazzy.

```
# Check service status
sudo systemctl status rovo_platform.service

# Start a service
sudo systemctl start rovo_platform.service

# Stop a service
sudo systemctl stop rovo_platform.service
```

Note: Service configurations may differ between Foxy and Jazzy installations.

Systemd Services

The ROVO2 uses systemd services for automatic startup of ROS2 nodes.

Install Startup Services:

```
ros2 run rovo_autostart startup_installer.py
```

Service Management:

Services can be managed via the webserver or command line:

```
# Check service status
sudo systemctl status rovo_platform.service

# Start a service
sudo systemctl start rovo_platform.service

# Stop a service
sudo systemctl stop rovo_platform.service

# Enable service at boot
sudo systemctl enable rovo_platform.service

# Disable service at boot
sudo systemctl disable rovo_platform.service
```

Available Services:

Service	Description
rovo_platform.service	Platform driver (CAN, odometry, IMU)
rovo_webserver.service	Web control interface
rovo_controller.service	Joystick and twist mux

Environment Configuration

Setup Script (config/setup.bash):

This script sets up environment variables and aliases.

Environment Variables:

Variable	Default	Description
ROVO_NS	rovo_unit_071	Robot namespace

Useful Aliases:

```
# Build and source workspace
rovo_build
```

CAN Interface Setup

The CAN interface is configured via the rovo_can.bash script:

```
sudo ip link set can0 down
sudo ip link set can0 type can bitrate 500000
sudo ip link set can0 up
```

udev Rules

Hardware udev rules are installed at /etc/udev/rules.d/45-mbs.rules for consistent device naming.

Files:

- config/setup.bash - Environment variables and aliases
- config/startup.bash - Startup script
- config/rovo_can.bash - CAN interface setup
- debian/45-mbs.rules - udev rules for hardware
- scripts/startup_installer.py - Systemd service installer

Webserver

> **Note:** Web interface is not available for ROS Noetic.

Robot Webserver

> **Note:** The webserver interface for ROS2 Foxy may have limited functionality compared to Jazzy. Please upgrade to ROS2 Jazzy for the full webserver experience.

Access: http://192.168.131.1:9000

Robot Webserver

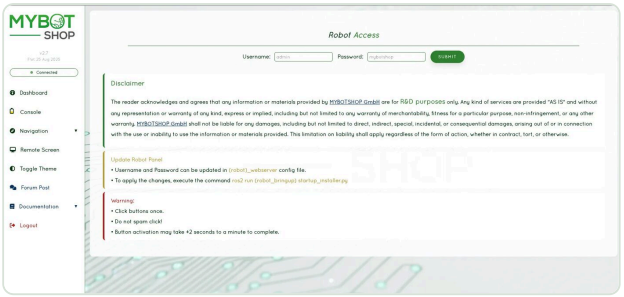
The ROVO2 features a web-based control interface built with Flask.

Access: http://192.168.131.1:9000

Default Credentials:

- Username: admin
- Password: mybotshop

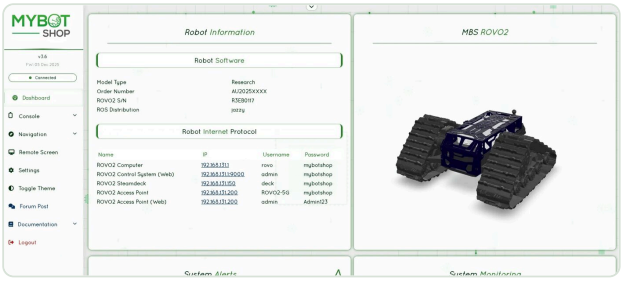
Login



_images/web_login.webp

Webserver Login Page

Dashboard



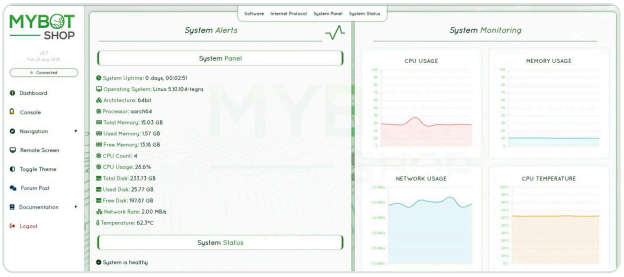
_images/web_dashboard.webp

Main Dashboard with 3D Robot Visualization

Features:

Feature	Description
3D Visualization	Real-time robot model via GLTF
Joystick Teleop	Browser-based joystick control
Service Management	Start/stop ROS2 nodes and services
Map Visualization	View navigation maps
Battery Monitoring	Real-time battery state
GPS Waypoints	Record and manage GPS waypoints
Rosbag Recording	Start/stop rosbag recordings
VNC Access	Remote desktop via VNC

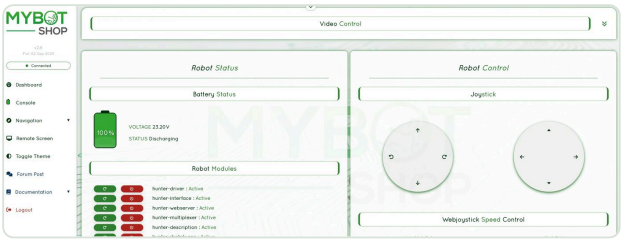
System Management



_images/web_system.webp

System Management Interface

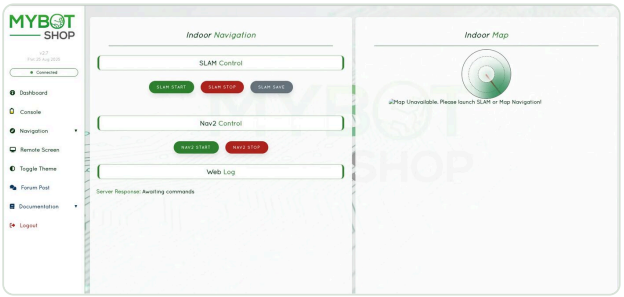
Console



_images/web_console.webp

Web Console Interface

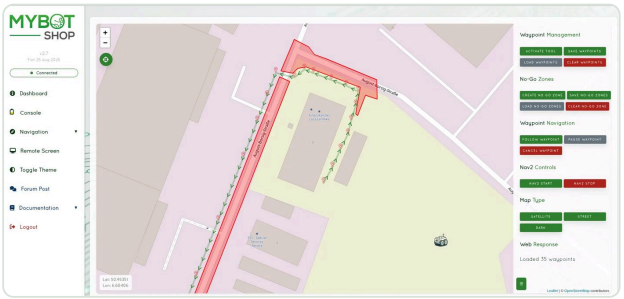
Indoor Navigation



_images/web_nav_indoor.webp

Indoor Navigation View

Outdoor Navigation



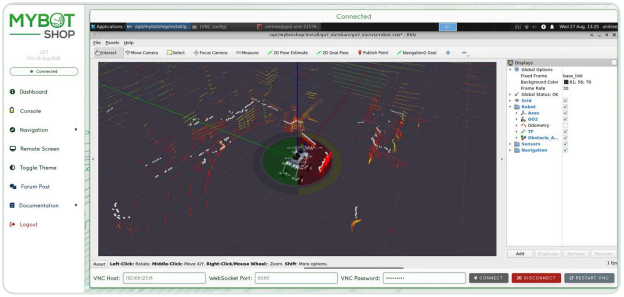
_images/web_nav_outdoor.webp

Outdoor Navigation View with GPS

VNC Remote Desktop

_images/web_vnc.webp

VNC Remote Desktop Access



_images/web_vnc_viz.webp

VNC with RViz2 Visualization

Configuration (config/robot_webserver.yaml):

Parameter	Description
robot_rosbag_dir	Rosbag storage directory
robot_services	List of controllable services
robot_map_topic	Map topic name
robot_cmd_vel	Velocity command topic
robot_e_stop	Emergency stop topic
robot_gps_topic	GPS topic
robot_battery_topic	Battery state topic

Launch Webserver:

The webserver is typically started automatically via systemd. To launch manually:

```
ros2 launch rovo_webserver webserver.launch.py
```

Dependencies:

```
pip3 install Flask playsound TTS
sudo apt-get install espeak-ng
```

VNC Setup:

To enable VNC remote desktop:

```
vncpasswd ~/.vnc/passwd
# Password: mybotshop
```

Firmware Updates:

The webserver supports firmware updates for the ROS2 workspace:

1. Navigate to the Firmware Update section
2. Upload a .zip or .tar.gz archive containing the new workspace
3. The system automatically: Creates a backup of the current workspace Extracts and replaces the old workspace Rebuilds using colcon build
4. Restart ROS2 services after update

Teleoperation

Keyboard Teleop

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Joystick Control

The ROVO2 supports joystick control via the Logitech controller. See the Packages section for controller mapping details.

Keyboard Teleop

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Joystick Control

The ROVO2 supports joystick control via the teleop_twist_joy package.

> **Note:** For ROS2 Foxy, joystick and twist mux configurations may differ from Jazzy. Please refer to the package documentation for Foxy-specific setup.

Command Line Interface

Teleoperate the ROVO2 using the keyboard via ROS2:

> **Warning:** Extreme caution required when using the ROVO2. Please go through the provided manuals before operating.

Prerequisites:

- Ensure rovo_autostart services are running (check via webserver)
- Robot platform drivers must be active
- Do not run duplicate drivers

1. Set Gear

Before moving the robot, set the appropriate gear:

```
# Set gear to 1 (drive mode)
ros2 service call /$ROVO_NS/platform/set_gear rovo_interface/srv/SetGear "{gear: 1}"
```

2. Launch Keyboard Teleop

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args --remap
cmd_vel:=/$ROVO_NS/cmd_vel
```

3. Control Keys

Key	Action
i	Move forward
k	Stop
,	Move backward
j	Turn left
l	Turn right
u	Forward + left
o	Forward + right
m	Backward + left
.	Backward + right
q/z	Increase/decrease speed

4. Return to Neutral

When finished, set the gear back to neutral:

```
ros2 service call /$ROVO_NS/platform/set_gear rovo_interface/srv/SetGear "{gear: 0}"
```

Joystick Control

The ROVO2 supports Logitech joystick control with the following button mapping:

Button	Function
Button 4 (LB)	Enable movement (hold)
Button 5 (RB)	Enable turbo mode
Left Stick Y	Linear X velocity
Right Stick X	Angular Z velocity

Speed Modes:

- Normal Mode: 0.2 m/s linear, 0.3 rad/s angular
- Turbo Mode: 1.0 m/s linear, 1.0 rad/s angular

Twist Mux Priority

The ROVO2 uses a twist multiplexer to handle velocity commands from multiple sources:

Priority	Source	Topic
255	E-Stop (Lock)	e_stop
25	Logitech Joystick	joy_teleop/cmd_vel
20	Steamdeck	steamdeck_joy_teleop/cmd_vel
15	Web Interface	webserver/cmd_vel
10	Interactive Marker	twist_marker_server/cmd_vel
5	External	cmd_vel
3	Autonomous (High)	autonomous_high_priority/cmd_vel
2	Autonomous (Mid)	autonomous_mid_priority/cmd_vel
1	Autonomous (Low)	autonomous_low_priority/cmd_vel

Web Interface Control

Access the webserver at http://192.168.131.1:9000 for browser-based joystick control.

Steamdeck Control

The ROVO2 can be controlled via Steamdeck using the steamdeck_joy_teleop interface.

Visualization

RViz

Launch RViz for robot visualization:

```
roslaunch rovo_viz view_robot.launch
```

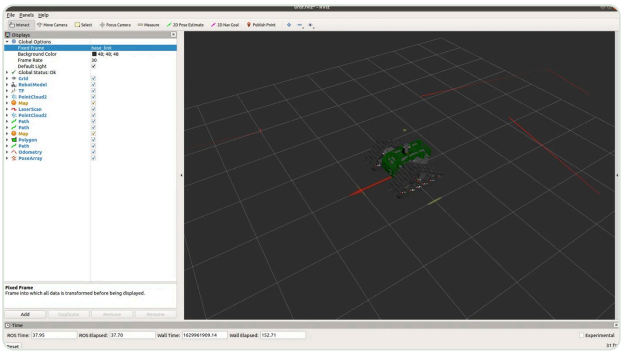
RViz2

Launch RViz2 for robot visualization:

```
ros2 launch rovo_viz view_robot.launch.py
```

RViz2

RViz2 is used for visualizing the robot state, sensor data, and navigation information.



_images/rovo_rviz_gazebo.webp

ROVO2 Visualization in RViz2 with Gazebo Simulation

Launch RViz2:

```
ros2 launch rovo_viz view_robot.launch.py
```

Available Configurations:

Configuration	Description
robot.rviz	Full robot visualization with navigation
model.rviz	Robot model only
steam.rviz	Steamdeck optimized view

Displayed Elements:

- Robot model (URDF)
- TF transforms
- Odometry
- LiDAR point cloud
- Depth camera images
- Navigation costmaps

- Path planning visualization

External Host Setup:

To visualize the robot from an external computer:

1. Build visualization packages: colcon build --symlink-install --packages-select rovo_description rovo_viz source install/setup.bash
2. Set the robot namespace: export ROVO_NS = "rovo_unit_071"
3. Launch RViz2: ros2 launch rovo_viz view_robot.launch.py

Note: Ensure your computer is on the same network as the robot and ROS2 DDS discovery is configured properly.

Navigation

Navigation Setup

Important: It is imperative to tune the Rovo according to your requirements before testing it. Be sure to remain vigilant when the robot is moving autonomously.

Hardware Requirements:

- Autonomous ground vehicle (AGV)
- Camera (Visual Odometry)
- Inertial measurement unit (IMU)
- Logitech controller
- Recommended: LiDAR, Depth Camera

Software Requirements:

- Ubuntu 18.04/Ubuntu 20.04
- ROS-melodic/ROS-noetic
- MBS Navigation Package

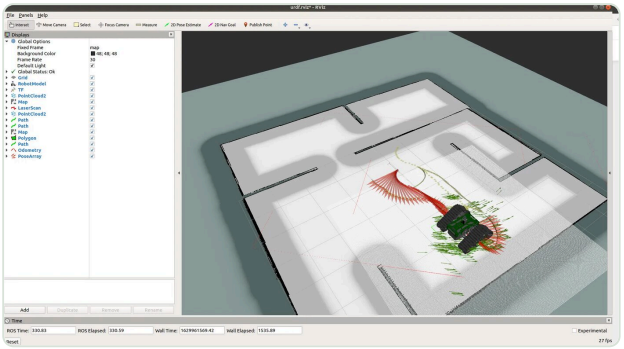
Warning: The provided MBS navigation package allows for point to point navigation utilizing GPS coordinates or indoor map coordinates. Caution must be exercised when using the package around people.

Simulation Navigation

1. Start the simulation and the navigation stack:

```
roslaunch rovo_gazebo gazebo_world.launch
roslaunch rovo_navigation odom_navigation.launch
```

1. Use the 2D pose estimate tool in RViz to correct position if needed.
2. Select the 2D Nav Goal to set a navigation target.



rovo_gazebo_completed_navigation_stack

Real-world Navigation

1. Start the navigation stack:

```
roslaunch rovo_base base.launch
roslaunch rovo_navigation odom_navigation.launch
```

1. Use RViz tools to set position and goals.

Mapping

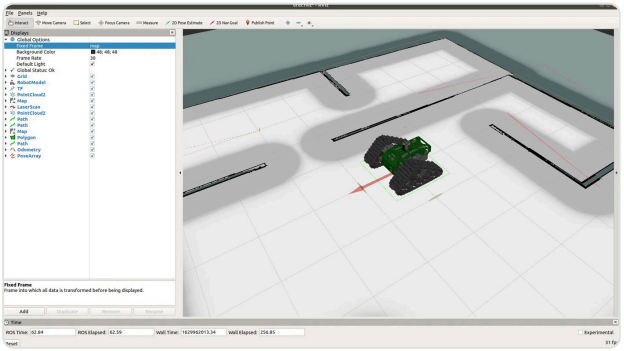
A sensor device is required with the Rovo for mapping such as a depth camera or a lidar.

Simulation Mapping:

```
roslaunch rovo_gazebo gazebo_world.launch
roslaunch rovo_navigation map_navigation.launch
rosrun key_teleop key_teleop.py key_vel:=rovo_velocity_controller/cmd_vel
```

Save the Map:

```
rosrun map_server map_saver -f rovo_map
```



rovo_gazebo_rviz_navigation_stack

Real-world Mapping:

```
rosrun gmapping slam_gmapping scan:=base_scan
```

Navigation

Nav2 navigation is supported on ROS2 Foxy.

Odometry Navigation:

```
ros2 launch rovo_navigation odom_navigation.launch.py
```

Map Navigation:

```
ros2 launch rovo_navigation map_navigation.launch.py
```

Note: Nav2 parameters and behavior trees may differ between Foxy and Jazzy. Please use the appropriate configuration files for your ROS2 version.

Simultaneous Localization and Mapping

SLAM allows the robot to build a map while localizing itself within it.

Prerequisites:

- Ensure rovo_autostart services are running
- LiDAR sensor must be active

Launch SLAM:

```
ros2 launch rovo_navigation slam.launch.py
```

Mapping Procedure:

1. Launch SLAM as shown above
2. Use teleop to drive the robot slowly (recommended: 0.2 m/s)
3. Cover all areas you want to map
4. Ensure good loop closures by revisiting previously mapped areas

Save the Map:

Once satisfied with the map, save it:

```
ros2 run nav2_map_server map_saver_cli -f
/opt/mybotshop/src/mybotshop/rovo_navigation/maps/custom_map \
  --ros-args --remap map:=/$ROVO_NS/map
```

Rebuild Workspace (if using custom map name):

```
cd /opt/mybotshop && colcon build --symlink-install
source /opt/mybotshop/install/setup.bash
```

Odometry Navigation

Odometry navigation allows the robot to navigate using only odometry data, without a pre-built map.

Launch Odometry Navigation:

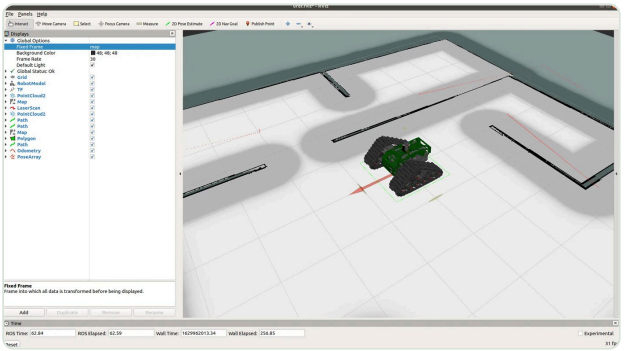
```
ros2 launch rovo_navigation odom_navi.launch.py
```

This mode is useful for:

- Simple point-to-point navigation
- Testing navigation in unknown environments
- Scenarios where mapping is not required

Map Navigation

Map-based navigation provides full autonomous navigation using a pre-built map.



_images/rovo_rviz_navigation_stack.webp

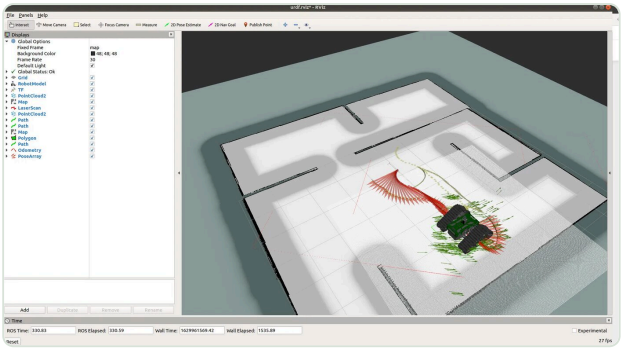
ROVO2 Navigation Stack in RViz2

Prerequisites:

- A saved map must exist
- Map must be built and available in the ROS package

Launch Map Navigation:

```
ros2 launch rovo_navigation map_navi.launch.py
```



_images/rovo_rviz_navigation_completed.webp

ROVO2 Navigation Goal Completed

Setting Goals:

Goals can be set via:

1. RViz2 “2D Goal Pose” tool
2. Command line
3. Webserver interface

Configuration Files:

- param/nav2_slam.yaml - SLAM parameters
- param/nav2_odom.yaml - Odometry navigation parameters
- param/nav2_map.yaml - Map navigation parameters
- behavior_trees/nav_to_pose.xml - Custom behavior tree

Dependencies:

```
sudo apt-get install ros-jazzy-navigation2 ros-jazzy-nav2-bringup ros-jazzy-twist-mux
```

For more information, see the Nav2 documentation .

GPS Navigation

GPS-based navigation is available for outdoor operations. Configure based on your GPS hardware and use the webserver for waypoint recording and playback.

Sensors

Sensor Augmentation

Rovo can be upgraded with different sensors and accessories.

Ouster LiDAR

Attention: The Ouster organization have updated their installation method. Please follow the instructions in the Ouster ROS github .

Dependencies:

```
sudo apt install build-essential cmake libglfw3-dev libglew-dev libeigen3-dev \
     libjsoncpp-dev libtclap-dev
```

ROS Dependencies:

```
sudo apt install ros-melodic-ros-core ros-melodic-pcl-ros \
     ros-melodic-tf2-geometry-msgs ros-melodic-rviz
```

ZED2 Camera

Requirements:

- Zed Camera
- CUDA 10.2+
- Internet connection

Verify Installation:

```
roslaunch zed_wrapper zed2.launch
roslaunch zed_display_rviz display_zed2.launch
```

Emlid REACH RS2 GPS

EMLID REACH RS2

Data Sheets:

- Reach Emlid RS2 Data Sheet

Documentation:

- Emlid RS2 Tutorial
- Emlid RS2 Base Rover Tutorial

Important: Please note that the Emlid Reach RS2 works on 2.4GHz band when connecting to a WiFi network. Position of GPS relative to the ground severely effects accuracy and connectivity.

Setup Procedure:

1. Connect to the Emlid via WiFi hotspot (SSID: Reach, Password: emlidreach)
2. Install the ReachView 3 app
3. Configure settings

ROS Driver:

```
cd catkin_ws/src
git clone https://github.com/enwaytech/reach_rs_ros_driver
catkin build
```

Phidgets IMU

_images/phidgets_imu.webp

Phidgets IMU

Check Status:

```
sudo service mbs_husky status
```

Check IMU Data:

```
rostopic echo /IMU/data_raw
rostopic echo /IMU/mag
```

Depth Cameras

Intel RealSense D435i is supported via the realsense-ros package for Foxy.

```
sudo apt-get install ros-foxy-realsense2-camera
```

Lidars

Ouster LiDAR support is available through the Ouster ROS2 driver.

> **Note:** Sensor drivers for ROS2 Foxy may have different configurations than Jazzy. Please check individual sensor documentation for Foxy compatibility.

Depth Cameras

The ROVO2 supports the Intel RealSense D435i depth camera for RGB-D perception.

Test Native Driver:

```
ros2 launch realsense2_camera rs_launch.py depth_module.depth_profile:=1280x720x30
pointcloud.enable:=true
```

Launch via ROVO2 Package:

The camera can be controlled via the webserver or systemctl. When disabled via services, test manually:

```
ros2 launch rovo_depth_camera realsense.launch.py
```

Published Topics:

Topic	Type
/camera/color/image_raw	sensor_msgs/Image
/camera/depth/image_rect_raw	sensor_msgs/Image
/camera/depth/color/points	sensor_msgs/PointCloud2

> **Note:** The ROVO2 utilizes the camera's inbuilt IMU. External USB connection may be required for depth functionality.

Lidars

The ROVO2 supports Ouster OS1-64 LiDAR for 3D perception and mapping.

_images/outdoor_pointcloud.webp

Ouster LiDAR Point Cloud Visualization

Network Configuration:

- Default IP: 192.168.131.20

Launch via ROVO2 Package:

The LiDAR can be controlled via the webserver or systemctl. When disabled via services, test manually:

```
ros2 launch rovo_lidar ouster.launch.py
```

Published Topics:

Topic	Type
/ouster/points	sensor_msgs/PointCloud2
/ouster/scan	sensor_msgs/LaserScan
/ouster/imu	sensor_msgs/Imu

Dependencies:

```
# Install Ouster ROS2 driver
sudo apt-get install ros-jazzy-ouster-ros
```

For more information, see the Ouster ROS2 driver documentation .

GPS

The ROVO2 supports the Emlid Reach RS2 for high-precision RTK GPS positioning.

_images/emlid.webp

Emlid Reach RS2 GPS Module

GPS integration is available for outdoor navigation and waypoint recording. Configuration depends on the specific GPS hardware installed on your ROVO2 unit.

Auxiliary

The ROVO2 platform includes an integrated IMU that publishes orientation and acceleration data.

Platform IMU Published Topic:

- /<ns>/platform/imu/data (sensor_msgs/Imu)

The ROVO2 also supports external Phidgets Spatial IMU for enhanced orientation accuracy.

_images/phidgets_imu.webp

Phidgets Spatial IMU

Installation:

```
sudo apt-get install ros-jazzy-phidgets-drivers
```

Published Topics:

Topic	Type
/imu/data	sensor_msgs/Imu
/imu/mag	sensor_msgs/MagneticField

The BMS provides battery state information.

Published Topic:

- /<ns>/platform/bms/state (sensor_msgs/BatteryState)

Simulation

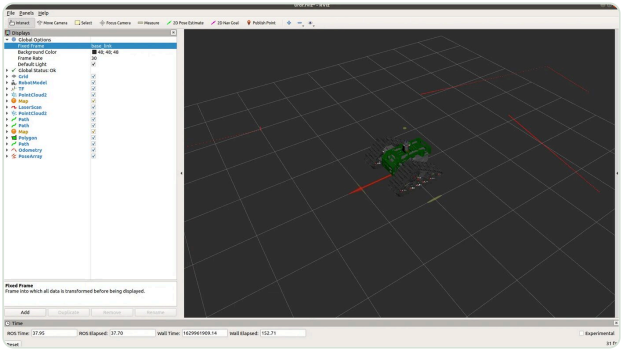
Gazebo Simulation

rovo_gazebo

Once the ROS packages are set up for use, you can verify whether they are working correctly or not by running the following command:

```
roslaunch rovo_simulator gazebo_world.launch
```

This should start two applications: RViz and Gazebo with pre-loaded gazebo drivers.



rovo_gazebo_rviz

Be sure to change the Fixed Frame in the global options to base_link .

Important: Due to simulation limitation the tracks of the Rovo do not make contact with the ground, instead the treads are directly into contact with the ground. Also, simulation with the Rovo is computationally heavy, so a powerful computer is required to run the simulations.

Gazebo

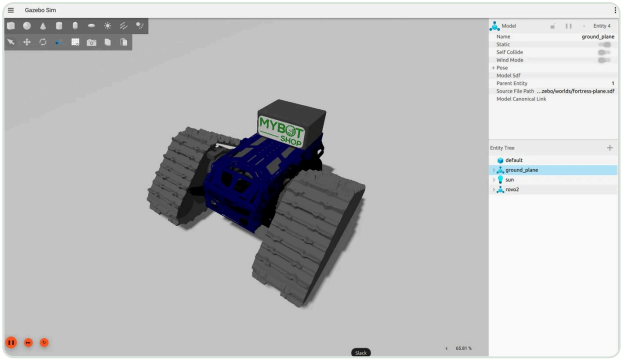
ROS2 Foxy uses Gazebo Classic (Gazebo 11) for simulation.

```
ros2 launch rovo_simulator gazebo_world.launch.py
```

Note: For ROS2 Jazzy, use Gazebo Harmonic instead of Gazebo Classic.

Gazebo

The ROVO2 uses Gazebo Harmonic for simulation, which is the recommended simulator for ROS2 Jazzy.



_images/rovo2_gazebo.webp

ROVO2 Simulation in Gazebo Harmonic

Launch Simulation:

```
ros2 launch rovo_gazebo simulation.launch.py
```

Available Worlds:

World File	Description
fortress-plane.sdf	Flat ground plane (default)
fortress-substation.sdf	Industrial substation environment
fortress-moon.sdf	Lunar terrain

Teleoperate in Simulation:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Simulated Sensor Topics:

Topic	Type	Description
/clock	rosgraph_msgs/Clock	Simulation time
/odom	nav_msgs/Odometry	Robot odometry
/sensors/imu/data	sensor_msgs/Imu	IMU data
/sensors/d435i/image	sensor_msgs/Image	RGB camera
/sensors/d435i/depth_image	sensor_msgs/Image	Depth image
/sensors/d435i/points	sensor_msgs/PointCloud2	RGB-D point cloud
/sensors/os_sensor/scan	sensor_msgs/LaserScan	LiDAR scan

/sensors/os_sensor/points	sensor_msgs/PointCloud2	LiDAR point cloud

Clean Up Gazebo Processes:

If Gazebo does not close properly, use the cleanup script:

```
ros2 run rovo_gazebo kill_gz.sh
```

Dependencies:

```
sudo apt-get install ros-jazzy-ros-gz ros-jazzy-ros-gz-bridge
```

For more information, see the Gazebo Harmonic documentation .

Isaac Sim

NVIDIA Isaac Sim integration is planned for future releases. Check back for updates on Isaac Sim support for the ROVO2 platform.

6 Safety Guidelines

Autonomous Mobile Robot

Deploying autonomous mobile robots mandates a focus on safety procedures to prevent accidents and guarantee secure operations. The subsequent guidelines delineate crucial safety measures for working with autonomous mobile robots.

Work Area Safety

- Ensure the workspace is tidy and adequately illuminated. Cluttered or dimly lit environments can hinder sensor performance and navigation systems.
- Refrain from deploying autonomous mobile robots in explosive settings, including areas with flammable substances like gases, liquids, or dust.
- Maintain a safe distance between bystanders and unauthorized individuals during robot operation to avert interference with autonomous navigation.

Electrical Safety

- Guarantee the robot's power system complies with electrical safety standards. Conduct routine inspections and upkeep of power components to avert malfunctions.
- Introduce safeguards to shield the robot from unfavorable weather conditions, like rain or damp environments.
- Perform regular assessments of power cables and connections, promptly replacing any damaged components to mitigate the risk of electrical complications.

Navigation Safety

- Introduce obstacle detection and avoidance systems to avert collisions with objects, individuals, or other robotic entities.
- Establish and uphold safety zones within the robot's operational vicinity to mitigate inadvertent interactions with personnel or other machinery.
- Conduct periodic calibration and testing of the robot's navigation sensors and systems to uphold precise and dependable functionality.

Emergency Response

- Integrate an emergency stop mechanism to swiftly cease the robot's operation during unforeseen circumstances or emergencies.
- Clearly identify and communicate emergency stop locations across the robot's operational vicinity.

- Organize routine emergency response drills to acquaint personnel with protocols for managing unexpected situations.

Data Security and Privacy

- Introduce robust cybersecurity protocols to safeguard the robot's control systems and data against unauthorized access or tampering.
- Uphold compliance with privacy regulations when acquiring, storing, or transmitting data gathered by the robot's sensors.
- Conduct regular updates of software and firmware to tackle security vulnerabilities and fortify the overall system integrity.

Human Interaction Safety

- Integrate sensors and communication systems capable of detecting and responding to human presence in the robot's proximity.
- Employ visual and audible signals to effectively convey the robot's operational status and intentions, ensuring nearby individuals are informed.
- Develop precise protocols for secure human-robot collaboration, particularly in shared workspaces, to maintain safety and efficiency in operations.

Residual Risks

Although safety measures are in place, residual risks may persist, including:

- Impairment of sensor functionality
- Potential collisions in crowded or dynamic environments
- Cybersecurity vulnerabilities
- Unintended human interactions due to unforeseen circumstances

Important: Autonomous mobile robots are advanced technologies that demand proper usage to evade accidents and maintain a secure environment. Learning and diligently adhering to correct procedures are crucial. Prioritizing both quality and safety is paramount. By adhering to these safety guidelines, you actively contribute to ensuring a safe environment during the deployment of autonomous mobile robots.

7 Platform Overview & Specifications

Introduction

_images/mbs_rovo2.webp

MYBOTSHOP ROVO2 Tracked Robot Platform

The MYBOTSHOP ROVO2 is a rugged tracked robot platform designed for autonomous mobile robot (AMR) applications. The tracked chassis provides superior terrain traversal capabilities compared to wheeled platforms, making it ideal for outdoor environments, rough terrain, and challenging surfaces.

The power pack is driven by 2 x 7.5KW electric motors, reduced via 1:16 gears providing considerable payload and tractive force. Despite the high gear reduction, the MBS ROVO2 reaches a top speed of 20km/h. The battery integrated in the chassis provides an extremely low center of gravity and a range of approximately 8 hours or 40km.

Quick Links

Resource	Link
MYBOTSHOP Website	mybotshop.de
ROVO2 ROS Documentation	docs.mybotshop.de
Robot Webserver	192.168.131.1:9000
ROS2 Jazzy Documentation	docs.ros.org/jazzy
Gazebo Harmonic Docs	gazebosim.org
Nav2 Documentation	nav2.org

External Dependencies

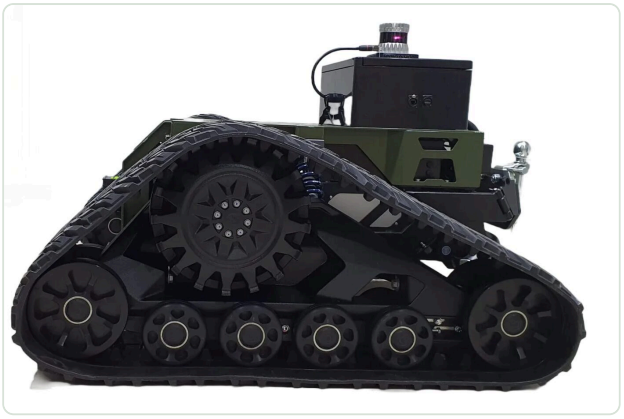
Package	Documentation
Ouster LiDAR Driver	ouster-ros
Intel RealSense ROS2	realsense-ros
Teleop Twist Keyboard	teleop_twist_keyboard
Nav2 Navigation	navigation2
ros_gz Bridge	ros_gz

Current Available Features

Feature	Status	Notes
ROVO2 Visualization (RViz2)	Available	
ROVO2 Base High Level Driver	Available	Joint State, IMU, CMD, Odometry
ROVO2 Description (URDF/Xacro)	Available	
ROVO2 Lidar (Ouster)	Planned	
ROVO2 Depth Camera (Intel Realsense D435i)	Planned	
ROVO2 Autostart Services	Available	
ROVO2 Twist Multiplexer	Available	
ROVO2 SLAM	Planned	
ROVO2 Odometry Navigation	Planned	
ROVO2 Map Navigation	Planned	
ROVO2 Webserver	Available	
ROVO2 Gazebo Harmonic Simulation	Available	

ROVO Platform

The ROVO2 is based on the HAWE Mattro ROVO platform, a fully electric tracked drive system designed for off-highway autonomous applications.



_images/rovo_side.webp

ROVO2 Side View

Manuals

> **Note:** Product documentation and manuals are available for customers only. Please contact MYBOTSHOP for access to the following documents.

Specifications

Mechanical

Parameter	Value
Chassis Type	Tracked
Drive System	Dual Track Drive (2 x 7.5KW)
Gear Ratio	1:16
Top Speed	20 km/h
IP Rating	IP65

Electrical

Parameter	Value
Battery Type	Lithium (Integrated)
Battery Range	~8 hours / 40km
Communication Interface	CAN Bus
CAN Bitrate	500 kbit/s

Base Software

Parameter	Value
Operating System	Ubuntu 24.04 LTS
ROS Distribution	ROS2 Jazzy
Simulator	Gazebo Harmonic
Navigation Stack	Nav2

Datasheets

Core Components

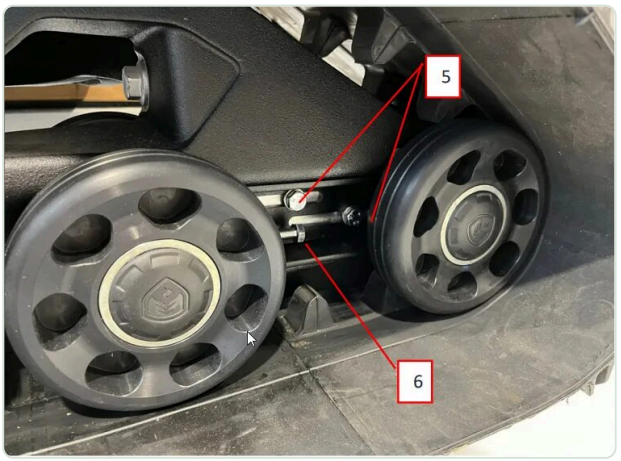
- MBS ROVO2 Data Sheet
- MBS ROVO2 User Manual EN
- MBS ROVO2 User Manual DE
- Phidgets Data Sheet
- SmartMOD UP6-500 Embedded Box PC Data Sheet
- SmartMOD UP6-500 Embedded Box PC Manual

Auxiliaries

- Zed2i Data Sheet
- Reach Emlid RS2 Data Sheet
- Ouster Data Sheets

Belt Tightening

In case the ROVO veers off to either the left or right, the issue may be traced back to the tracks being loose.



_images/rovo_belt.webp

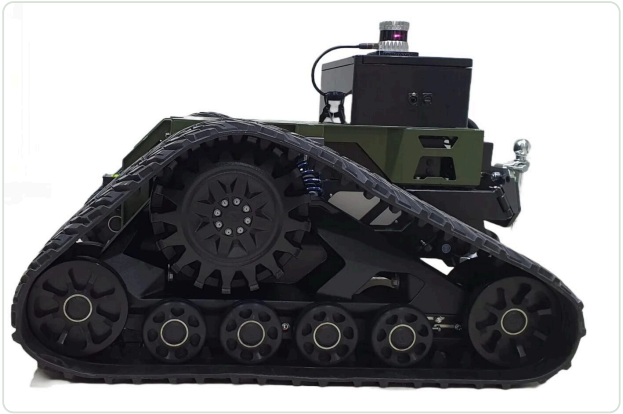
ROVO2 Belt Tightening Adjustment

To remedy this situation:

1. First loosen screw number 5
2. Adjust the track tension with screw number 6
3. Finally, tighten screw number 5 again with 25Nm

8 ROS 2 Package & API Reference

Packages



_images/rovo_side.webp

A rospackage that contains the core functionalities to enable the robot to function autonomously.

rovo_base

This ros package contains the hardware drivers that communicate with the 3-phase motors of the Rovo.

```
roslaunch rovo_base base.launch
```

rovo_bringup

This ros package launches the base driver as well as all necessary auxiliary components such as the inertial measurment unit (IMU), Lidar (if availabe), robot description (required for visualization), and ekf_localization (robot position estimation).

```
roslaunch rovo_bringup system_bringup.launch
```

rovo_control

This ros package fuses the robots tracked wheels movement with the inertial measurment unit (IMU) using the extended kalman filter. It provides the transforms from the robots base link to the odometry frame as well as mapping for the logitech controller.

Keys	Description
1	Switches Rovo to higher gear
2	Switches Rovo to lower gear
9	Enables control of the ROVO via ROS
5	Activates the horn in Rovo

j	Provides acceleration for in the direction pushed in Rovo

rovo_description

This is a ros package that contains the 3D models of the Rovo in URDF format.

Warning: Running this command will shut the Rovo Bringup description driver.

```
roslaunch rovo_description description.launch
```

rovo_navigation

Odom Navigation:

```
roslaunch rovo_navigation odom_navigation.launch
```

Map Navigation:

```
roslaunch rovo_navigation map_navigation.launch
```

rovo_simulator

```
roslaunch rovo_simulator gazebo_world.launch
```

rovo_viz

This ros package is used for visualization of the robots current state.

```
roslaunch rovo_viz view_robot.launch
```

rovo_description

URDF and Xacro robot description files for the ROVO2 tracked robot platform.

```
ros2 launch rovo_description description.launch.py
```

rovo_base

Hardware driver for the ROVO2 platform.

```
ros2 run rovo_base base_driver
```

rovo_bringup

System bringup package.

```
ros2 launch rovo_bringup bringup.launch.py
```

rovo_viz

Visualization package for RViz2.

```
ros2 launch rovo_viz view_robot.launch.py
```

Note: ROS2 Foxy packages are legacy. For the latest features, use ROS2 Jazzy.

rovo_description

URDF and Xacro robot description files for the ROVO2 tracked robot platform.

Contents:

- launch/rovo_description.launch.py - Robot state publisher
- launch/view_robot.launch.py - View URDF in RViz
- meshes/ - Platform and sensor visual meshes
- xacro/robot.xacro - Main robot description

Usage:

```
# Publish robot description
ros2 launch rovo_description rovo_description.launch.py

# Get URDF from Xacro
xacro $(ros2 pkg prefix rovo_description)/share/rovo_description/xacro/robot.xacro
```

rovo_platform

Hardware driver and control interface for the ROVO2 tracked robot platform. Provides CAN bus communication, motor control, odometry publishing, and ROS2 service interfaces.

Configuration Parameters (config/rovo_platform.yaml):

Parameter	Default	Description
node_id	2	CAN node ID
can_channel	can0	CAN interface name
bitrate	500000	CAN bitrate (500 kbit/s)
wheel_separation	0.92	Distance between tracks (m)
max_speed_mps	1.0	Maximum linear speed (m/s)
drive_diameter	0.35	Drive sprocket diameter (m)
gear_ratio	16.0	Gear ratio (16.0 for i16, 7.0 for i7)

publish_tf	true	Publish TF transforms

Published Topics:

Topic	Type	Description
/<ns>/platform/odom	nav_msgs/Odometry	Robot odometry
/<ns>/platform/joint_states	sensor_msgs/JointState	Track joint states
/<ns>/platform/imu/data	sensor_msgs/Imu	IMU data
/<ns>/platform/bms/state	sensor_msgs/BatteryState	Battery state

Subscribed Topics:

Topic	Type	Description
/<ns>/platform/cmd_vel	geometry_msgs/Twist	Velocity commands

Services:

Service	Type	Description
/<ns>/platform/set_gear	rovo_interface/SetGear	Set gear (0=neutral, 1-3=drive)

Usage:

```
ros2 launch rovo_platform rovo_driver.launch.py

# Set gear
ros2 service call /$ROVO_NS/platform/set_gear rovo_interface/srv/SetGear "{gear: 1}"
```

rovo_controller

Joystick teleoperation and command multiplexing for the ROVO2 platform.

Twist Mux Priority Table:

Priority	Source	Topic
255	E-Stop (Lock)	e_stop
25	Logitech Joystick	joy_teleop/cmd_vel
20	Steamdeck	steamdeck_joy_teleop/cmd_vel
15	Web Interface	webserver/cmd_vel
10	Interactive Marker	twist_marker_server/cmd_vel
5	External	cmd_vel

3	Autonomous (High)	autonomous_high_priority/cmd_vel
2	Autonomous (Mid)	autonomous_mid_priority/cmd_vel
1	Autonomous (Low)	autonomous_low_priority/cmd_vel

Logitech Joystick Button Mapping:

Button	Function
Button 4 (LB)	Enable movement
Button 5 (RB)	Enable turbo mode
Left Stick Y	Linear X velocity
Right Stick X	Angular Z velocity

Speed Settings:

Mode	Linear (m/s)	Angular (rad/s)
Normal	0.2	0.3
Turbo	1.0	1.0

rovo_gazebo

Gazebo Harmonic simulation package for the ROVO2 platform.

Available Worlds:

- fortress-plane.sdf - Flat ground plane (default)
- fortress-substation.sdf - Industrial substation environment
- fortress-moon.sdf - Lunar terrain

Simulated Topics (GZ -> ROS):

Topic	Type	Description
/clock	rosgraph_msgs/Clock	Simulation time
/odom	nav_msgs/Odometry	Robot odometry
/sensors/imu/data	sensor_msgs/Imu	IMU data
/sensors/d435i/image	sensor_msgs/Image	RGB camera
/sensors/d435i/depth_image	sensor_msgs/Image	Depth image
/sensors/d435i/points	sensor_msgs/PointCloud2	RGB-D point cloud
/sensors/os_sensor/scan	sensor_msgs/LaserScan	LiDAR scan

/sensors/os_sensor/points	sensor_msgs/PointCloud2	LiDAR point cloud

rovo_navigation

Nav2-based navigation stack for the ROVO2 platform.

Launch Files:

Launch File	Description
slam.launch.py	SLAM mapping mode
odom_navi.launch.py	Odometric navigation (no map)
map_navi.launch.py	Map-based autonomous navigation

Configuration Files:

- param/nav2_slam.yaml - SLAM parameters
- param/nav2_odom.yaml - Odometry navigation parameters
- param/nav2_map.yaml - Map navigation parameters
- behavior_trees/nav_to_pose.xml - Custom behavior tree

rovo_viz

RViz2 visualization package with pre-configured displays.

RViz Configurations:

- rviz/robot.rviz - Full robot visualization with navigation
- rviz/model.rviz - Robot model only
- rviz/steam.rviz - Steamdeck optimized view

rovo_autostart

System bringup and automatic startup configuration.

Files:

- config/setup.bash - Environment variables and aliases
- config/startup.bash - Startup script
- config/rovo_can.bash - CAN interface setup
- debian/45-mbs.rules - udev rules for hardware
- scripts/startup_installer.py - Systemd service installer

Environment Variables:

Variable	Default	Description
ROVO_NS	rovo_unit_071	Robot namespace

rovo_webserver

Web-based robot control interface with Flask backend.

Features:

- Real-time robot visualization (3D model via GLTF)
- Joystick teleoperation
- Service management (start/stop ROS nodes)
- Map visualization
- Battery monitoring
- GPS waypoint recording
- Rosbag recording
- VNC remote desktop access

Access: http://192.168.131.1:9000

rovo_interface

Custom ROS2 service definitions for the ROVO2 platform.

SetGear.srv:

```
uint8 gear      # 0=neutral, 1-3=drive gears
---
string message
bool success
```

Usage:

```
# Set gear to 1
ros2 service call /$ROVO_NS/platform/set_gear rovo_interface/srv/SetGear "{gear: 1}"
```

Debugging

Topic Debugging

List all topics:

```
rostopic list
```

Echo a topic:

```
rostopic echo /odom
```

Check topic frequency:

```
rostopic hz /odom
```

Topic Debugging

List all topics:

```
ros2 topic list
```

Echo a topic:

```
ros2 topic echo /odom
```

Check topic frequency:

```
ros2 topic hz /odom
```

Service Debugging

List all services:

```
ros2 service list
```

Call a service:

```
ros2 service call /rovo_base/set_gear rovo_msgs/srv/SetGear "{gear: 1}"
```

RQT

RQT provides a GUI framework for ROS2 debugging tools.

Launch RQT:

```
ros2 run rqt_gui rqt_gui --ros-args --remap tf:=/$ROVO_NS/tf --remap
tf_static:=/$ROVO_NS/tf_static
```

Available Plugins:

- Topic Monitor
- Service Caller
- Parameter Reconfigure
- Node Graph
- TF Tree

Dynamic Reconfigure

Use dynamic reconfigure to adjust parameters at runtime:

```
ros2 run rqt_gui rqt_gui --ros-args --remap tf:=/$ROVO_NS/tf --remap
tf_static:=/$ROVO_NS/tf_static
```

From the GUI, select Plugins > Configuration > Dynamic Reconfigure .

TF

TF Tree Visualization:

```
ros2 run rqt_tf_tree rqt_tf_tree --force-discover \
  --ros-args --remap tf:=/$ROVO_NS/tf --remap tf_static:=/$ROVO_NS/tf_static
```

Generate TF Frames PDF:

```
ros2 run tf2_tools view_frames.py --force-discover \
  --ros-args --remap tf:=/$ROVO_NS/tf --remap tf_static:=/$ROVO_NS/tf_static
```

Topic Debugging

List all topics:

```
ros2 topic list
```

Echo a topic:

```
ros2 topic echo /$ROVO_NS/platform/odom
```

Check topic frequency:

```
ros2 topic hz /$ROVO_NS/platform/odom
```

Service Debugging

List all services:

```
ros2 service list
```

Call a service:

```
ros2 service call /$ROVO_NS/platform/set_gear rovo_interface/srv/SetGear "{gear: 0}"
```

Node Debugging

List all nodes:

```
ros2 node list
```

Get node info:

```
ros2 node info /rovo_platform
```

CAN Debugging

Test CAN Module:

```
# Configure CAN interface
sudo ip link set can0 down
sudo ip link set can0 type can bitrate 500000
sudo ip link set can0 up

# Monitor CAN traffic
sudo candump can0
```

Miscellanious

Useful Commands

Build Workspace:

```
cd ~/catkin_ws
catkin build
source devel/setup.bash
```

Useful Commands

Build Workspace:

```
cd ~/ros2_ws
colcon build --symlink-install
source install/setup.bash
```

Check ROS2 Environment:

```
printenv | grep ROS
```

Sync Host and Robot

Synchronize files between host computer and robot using rsync:

Sync to Robot:

```
rsync -avP -t --delete -e ssh src robot@192.168.131.1://opt/mybotshop
```

Sync to Steamdeck:

```
rsync -avP -t --delete -e ssh src deck@192.168.131.150://home/deck/ros2_ws
```

Core Launch Files

Warning: Do not run these files manually as they should already be running via the webserver!

These launch files are managed by systemd services and the webserver:

Platform Driver:

```
ros2 launch rovo_platform rovo_driver.launch.py
```

Webserver:

```
ros2 launch rovo_webserver webserver.launch.py
```

Controller:

```
ros2 launch rovo_controller controller.launch.py
```

Robot Description:

```
ros2 launch rovo_description rovo_description.launch.py
```

CAN Hardware Connections

Recommended Parts:

- Primary: Part 1-1564337-1
- Backup: Part 967650-1

Pin Mapping:

ROVO	Peak-USB
Pin1 Can2 Low	Pin2 Can Low
Pin2 Can2 High	Pin7 Can High
Pin3 Can Gnd	Pin3 Can Gnd

Useful Commands

Build Workspace:

```
cd /opt/mybotshop
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
source install/setup.bash
```

Quick Build Alias:

```
rovo_build
```

Check ROS2 Environment:

```
printenv | grep ROS
```

Source Workspace:

```
source /opt/mybotshop/install/setup.bash
```

9 Legacy Quick Start – Rovo Barkhausen (ROS Noetic / ROS 2 Foxy)

1. Robot Interface

Guidelines for connecting with the robot Applicable on Ubuntu 20.04

1.1. Static Network Connection

For the initial setup, connecting via a LAN cable is necessary to configure the robot's WLAN network. To create a static connection on your PC (not the robot) in Ubuntu:

1. Navigate to Settings → Network, then click on + to generate a new connection.
2. Adjust the IPv4 settings to Manual.
3. Configure the IP Address as 192.168.126.51 and the Netmask as 24.
4. Save the settings and restart your network.

Once connected, verify the host's local IP using the following command in the Host PC's terminal:

```
ifconfig
```

Subsequently, perform a ping test to the robot:

```
ping 192.168.126.1
```

To access the robot via SSH, use the command:

```
ssh -X administrator@192.168.126.1
```

Please note, the default password is:

```
mybotshop
```

> **Note:** Sometimes other networks can cause disruptions when connecting to the Rovo. It is best to have only your connection to the robot active and all others inactive.

1.2. Screen Connection

Another option to connect to the robot involves using an HDMI cable along with a mouse and keyboard. This setup facilitates connecting the robot within your local network.3

2. Quick-Start ROVO2 (Real-Hardware)

2.1 Switching ROS Distributions

By default, the system operates on ROS Noetic. To transition to ROS Foxy, input the command:

```
foxy
```

To revert to ROS Noetic:

```
noetic
```

Running the appropriate ROS distribution is essential while utilizing the robot's ROS drivers.

2.2 ROVO2 Operation

Initiate the Hardware Launch for rovo by executing the following commands:

```
roslaunch rovo_bringup system_startup.launch
```

for ros2

```
ros2 launch rovo_bringup system_startup.launch.py
```

Adjust Gear Settings (1-4):

```
rosservice call /rovo_base/set_gear "gear: 1"
```

for ros2

```
ros2 service call /rovo_base/set_gear rovo_msgs/srv/SetGear "gear: 1"
```

These commands facilitate the launch of hardware components and allow for setting the gear value to a specified range (1-4) within the robot's system. The provided examples demonstrate the commands for both ROS and ROS 2 environments, ensuring compatibility and flexibility with different ROS versions. Adjustments in gear settings are performed through ROS services to regulate the robot's functionalities.

Note: When turning off the ROVO ros driver, please switch to gear 0 and then kill the driver node.

2.3 ROVO2 Remote

Once the ROS driver is active in either ROS1 or ROS2, you can manage the robot using the default remote control, offering ROS-based capabilities:

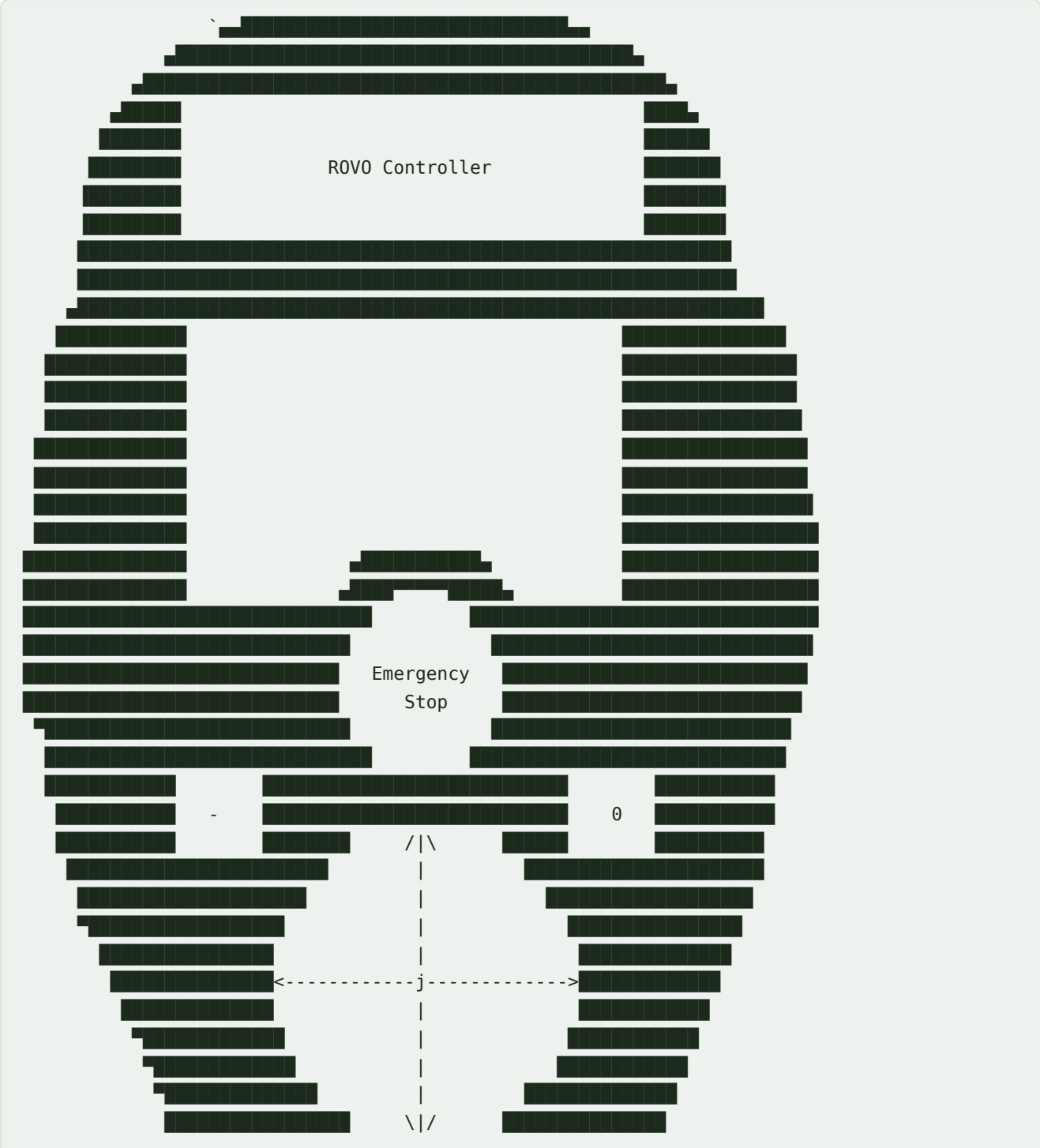
1. Button 1: Upshifts to a higher gear.
2. Button 2: Downshifts to a lower gear.
3. Button 3: Engages the horn.
4. Button 4: Activates the light switch.

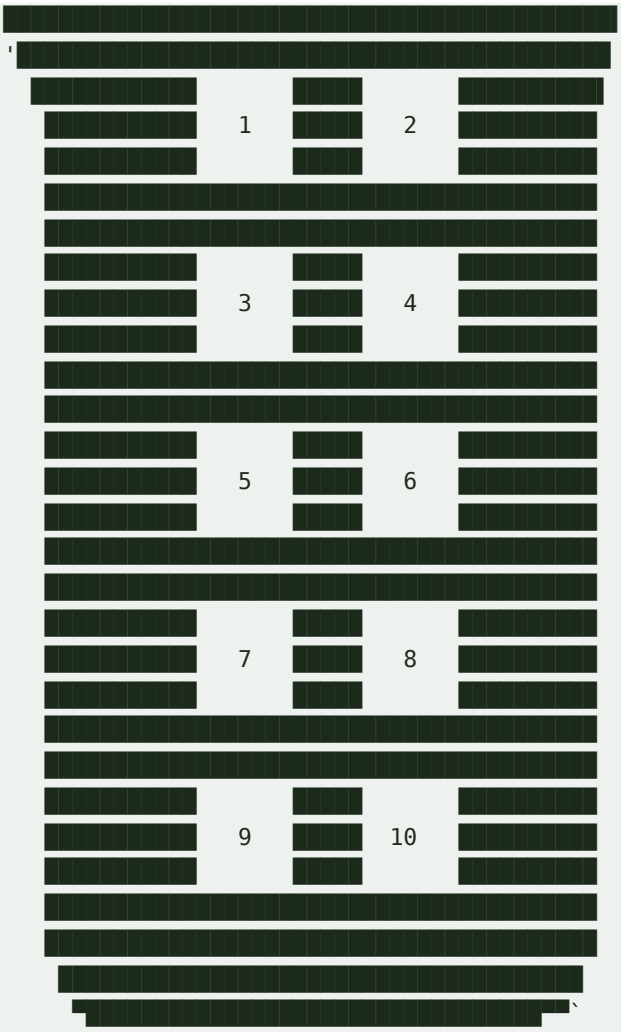
5. Button 9: Functions as the dead man's switch.
6. Joystick: Controls the ROVO's movement.

To maneuver the ROVO:

1. Select gear 1 or higher.
2. Press and hold Button 9.
3. Utilize the Joystick for movement control.

These instructions enable ROVO manipulation through the remote control, specifying the functions associated with each button and joystick action while ensuring proper maneuvering protocol for the robot.





Note: If either the gear is not set or Button 9 is not held than the robot will not move with the ROS driver.

2.4 Teleop Control (Real-Hardware)

To maneuver the robot, employ the teleop twist command:

For ROS:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

For ROS 2:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

These commands enable robot control using teleoperation twist functionalities, facilitating movement and navigation via the keyboard.

Note: Remember the gear should be 1 or higher for the robot to move, Also it is recommended that the robot moves in 0.1m/s.

2.5 Start Viz (Real-Hardware)

To observe the robot and its accessible sensors, execute the following file:

For ROS:

```
roslaunch rovo_viz view_robot.launch
```

For ROS 2:

```
ros2 launch rovo_viz view_robot.launch.py
```

This command initiates a file that provides a visual representation of the robot along with its associated sensors, allowing for convenient monitoring and assessment.

2.6 Start ROVO2 Odom Navigation (ROS1 Only)

To engage autonomous navigation, execute the subsequent launch file:

```
roslaunch rovo_navigation odom_navigation.launch
```

Important: Ensure the gear is set to 1 or higher and have the remote control nearby. To halt movement, press Button 9 on the ROVO Controller. Note that the controller takes precedence over velocity commands in the Navigation stack. Therefore, pressing Button 9 stops the robot's movement due to its higher priority in commanding velocities.

2.7 Enable Lights (Real-Hardware)

Execute the following commands to activate the robot's light:

For ROS:

```
rosservice call /rovo_base /set_light "data: true"
```

For ROS 2:

```
ros2 service call /rovo_base/set_light std_srvs/srv/SetBool data:\ true
```

These commands trigger the robot's light functionality, enabling it for use.

2.8 Enable Horn (Real-Hardware)

To activate the robot's horn, execute the following commands:

For ROS:

```
rosservice call /rovo_base/set_horn "data: true "
```

For ROS 2:

```
ros2 service call /rovo_base/set_horn std_srvs/srv/SetBool data :\ true
```

These commands trigger the robot's horn functionality, activating it for use.

3. Quick-Start Gazebo (Simulated-Hardware)

To launch the Gazebo world simulation for the ROVO robot, use the following command:

```
roslaunch rovo_simulator gazebo_world.launch
```

This command initializes the Gazebo simulation environment for the ROVO robot, allowing you to interact with and test its functionalities within the simulated world.

3.1 Teleop Control (Simulated-Hardware)

To maneuver the robot, utilize the teleop twist command:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Ensure the command '/rovo_velocity_controller/cmd_vel' is designated for the movement commands (cmd_vel). Adjust this command based on your system setup to enable robot control through teleoperation twist functionalities.

4. ROS Multi Machine (ROS1 Only)

Enables viewing the latency of the robot. The robot is set to the localhost by default.

4.1 MultiMachine - Host

To ensure proper communication across terminals in the robot (ideally 4-5 terminals), export the specified environment variables:

```
export ROS_MASTER_URI=http://192.168.126.1:11311/
export ROS_IP=192.168.126.1
export ROS_HOSTNAME=192.168.126.1
```

These commands enable seamless communication and coordination among the terminals within the robot.

4.2 Multimachine - Client To set up the environment variables across all terminals on your PC, use the following commands:

```
export ROS_MASTER_URI=http://192.168.126.1:11311/
export ROS_IP=192.168.126.52
export ROS_HOSTNAME=192.168.126.52
```

These commands ensure consistent communication across your PC's terminals by defining the ROS master URI, IP address, and hostname.

4.3 Multimachine Usage To initiate on the robot:

```
roslaunch rovo_bringup system_startup.launch
```

For the user's PC:

```
roslaunch rovo_viz view_robot.launch
```

These commands launch the necessary setups to start the robot's functionalities on one system and enable visualization on the user's PC.

5. Autonomous Mobile Robot Safety Guidelines

Deploying autonomous mobile robots mandates a focus on safety procedures to prevent accidents and guarantee secure operations. The subsequent guidelines delineate crucial safety measures for working with autonomous mobile robots:

5.1 Work Area Safety

⦿ Ensure the workspace is tidy and adequately illuminated. Cluttered or dimly lit environments can hinder sensor performance and navigation systems.

⦿ Refrain from deploying autonomous mobile robots in explosive settings, including areas with flammable substances like gases, liquids, or dust. The robot's components could present hazards in such environments.

⦿ Maintain a safe distance between bystanders and unauthorized individuals during robot operation to avert interference with autonomous navigation.

5.2 Electrical Safety

⦿ Guarantee the robot's power system complies with electrical safety standards. Conduct routine inspections and upkeep of power components to avert malfunctions.

⦿ Introduce safeguards to shield the robot from unfavorable weather conditions, like rain or damp environments.

⦿ Perform regular assessments of power cables and connections, promptly replacing any damaged components to mitigate the risk of electrical complications.

5.3 Navigation Safety

⦿ Introduce obstacle detection and avoidance systems to avert collisions with objects, individuals, or other robotic entities.

⦿ Establish and uphold safety zones within the robot's operational vicinity to mitigate inadvertent interactions with personnel or other machinery.

⦿ Conduct periodic calibration and testing of the robot's navigation sensors and systems to uphold precise and dependable functionality.

5.4 Emergency Response

⦿ Integrate an emergency stop mechanism to swiftly cease the robot's operation during unforeseen circumstances or emergencies.

⦿ Clearly identify and communicate emergency stop locations across the robot's operational vicinity.

⦿ Organize routine emergency response drills to acquaint personnel with protocols for managing unexpected situations.

5.5 Data Security and Privacy

⦿ Introduce robust cybersecurity protocols to safeguard the robot's control systems and data against unauthorized access or tampering.

⦿ Uphold compliance with privacy regulations when acquiring, storing, or transmitting data gathered by the robot's sensors.

⦿ Conduct regular updates of software and firmware to tackle security vulnerabilities and fortify the overall system integrity.

5.6 Human Interaction Safety

⦿ Integrate sensors and communication systems capable of detecting and responding to human presence in the robot's proximity.

⦿ Employ visual and audible signals to effectively convey the robot's operational status and intentions, ensuring nearby individuals are informed.

⦿ Develop precise protocols for secure human-robot collaboration, particularly in shared workspaces, to maintain safety and efficiency in operations.

5.7 Residual Risks

Although safety measures are in place, residual risks may persist, including:

⦿ Impairment of sensor functionality.

⦿ Potential collisions in crowded or dynamic environments.

⦿ Cybersecurity vulnerabilities.

⦿ Unintended human interactions due to unforeseen circumstances.

Important: Autonomous mobile robots are advanced technologies that demand proper usage to evade accidents and maintain a secure environment. Learning and diligently adhering to correct procedures are crucial. Prioritizing both quality and safety is paramount. By adhering to these safety guidelines, you actively contribute to ensuring a safe environment during the deployment of autonomous mobile robots.

10 Getting Help & RMA

RMA / Support

How to Raise a Return Merchandise Authorization (RMA) Case

At MYBOTSHOP GmbH, we are committed to providing excellent support to ensure your issues are resolved promptly. If you encounter any problems with our products, please follow the steps below to raise a Return Merchandise Authorization (RMA) case:

Step 1: Open a Topic in Our Forum

1. Visit MYBOTSHOP Forum .
2. Open a new topic describing your issue in detail. Our support team and community members will attempt to assist you in resolving the problem remotely.

Step 2: Contact Support for RMA Number

If the issue cannot be resolved remotely through our forum:

1. Email us at support @ mybotshop . de .
2. Provide a detailed description of the issue and include any troubleshooting steps already taken.
3. Our support team will evaluate your case and, if necessary, issue a RMA number along with a return address or a shipping label.

Important Shipping Information

Note: Please note: All goods must be sent back to the following address: MYBOTSHOP GmbH Willy - Messerschmitt - Strasse 12 50126 Bergheim Germany Do not ship goods without obtaining a RMA number. Parcels sent without a RMA number will be declined and returned to the sender.

Help Resources

In case of any issues, help can be taken from any of the following forums/resources:

1. MYBOTSHOP Forum : The forum is actively monitored by support teams at MYBOTSHOP GmbH .
2. Github Issues : Issues can be reported in the repository.
3. Online QA Sites: Questions can be asked on any of the QA sites like StackOverflow and Answers ROS .