MYBOTSHOP

QUADRUPED ROBOTICS

Unitree Go1

User Manual

CONTACT

info@quadruped.de

support@quadruped.de

quadruped.de

Contents

1 Attention

Attention: Read this document carefully. In case of doubt, it is essential to consult specialists, experts, or the manufacturer (Unitree Robotics) or distributor (MYBOTSHOP / QUADRUPED Robotics) of the assemblies used. The robot must not be put into operation before clarification. Contact information is available in Chapter 14, Getting Help & RMA.

Do not allow persons who are not familiar with the robotics or these instructions to operate the robot. Robots are dangerous in the hands of untrained users.

Do not operate the robot if you are intoxicated, under the influence of drugs, or unable to concentrate. Familiarize yourself with the low-level joint control safety implications and the emergency stop / damping procedure before switching the robot on.

Caution: Low-level (per-joint Position/Velocity/Torque) control disables the built-in balancing safeguards entirely — an unsupported GO1 in low-level mode will fall over. Always suspend the robot on a gantry/frame before testing low-level control. If equipped with the ROBOTIS OpenManipulator arm or a RoboSense LiDAR, review the GO1 & Open Manipulator Safety Guidelines (Chapter 13) before operating.

2 Disclaimer

The GO1 robot documented here is sold and supported by *MYBOTSHOP / QUADRUPED Robotics* as the German distributor for Unitree Robotics. Basic ROS 1 (Melodic/Noetic) understanding is required to operate the software stack covered in this manual — if you are not familiar with ROS, we recommend checking the ROS documentation first.

The client acknowledges and agrees that any information or materials provided by MYBOTSHOP / QUADRUPED Robotics 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 product is intended for civilian use only. Users should avoid making dangerous modifications or using the robot in hazardous ways. Some features described in this manual (e.g. the ROBOTIS OpenManipulator arm variant, RoboSense LiDAR, GPS/RTK localization) apply only to specific hardware configurations — always confirm which accessories are fitted to your unit before following an accessory-specific procedure. For terms, policies and compliance with local laws and regulations, refer to the Unitree Robotics website and the RMA/support process in Chapter 14.

MYBOTSHOP / QUADRUPED Robotics 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 GO1

GO1 Quadruped Tutorials

GO1 Robot

This package supplies Sphinx-based tutorial content to assist you with setting up and operating your GO1 quadruped robot. The tutorials topics are listed in the left column, and presented in the suggested reading order.

Key Features

- Fastest running speed of 4.7m/s (10.5mph, close to the speed of professional long-distance runners)
- ISS intelligent accompanying system, patented wireless vector positioning and control technology
- vSSS super sense system, 5 groups of fisheye binocular depth perception + fisheye AI perception + 3 groups of ultrasonic
- Built-in super AI computing power with 16-core top CPU + GPU (384Core, 1.5TFLOPS)

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

Simulation is a logical place for most users to start, as this is universally applicable; understanding how to effectively operate GO1 in simulation is valuable whether you are in the testing phase with software you intend to ultimately deploy on a real GO1, or you do not have one and are simply exploring the platform's capabilities.

Navigation is a follow-on to what is learned in the simulation tutorial, as navigation and map-making may be run in the simulated environment. However, this content is applicable to both the simulator and the real platform, if equipped with a laser scanner or lidar.

The remainder of the subjects are more applicable to the real robot, and have to do with configuring, using, and maintaining the platform. If you are a lab administrator rather than direct platform user, you may wish to skip the introductory chapters and jump straight to these ones.

4 Quick-Start & Network Setup

Quick-Start

Quadruped Robotics Europe is a young, but fast growing startup in the field of legged robotics. In this documentation, the software overview of GO1 robot is shown. The software is currently under development. At the moment simples ros drivers are available for the robot along with simulation are available in Gazebo.

As several variants of GO1 are provided with different hardware versions, the following video demonstrates operation of the software development kit GO1 Operation for the GO1.

Quick-Start Teleop

A pre-configured GO1 can start its navigation via the following commands:

Attention: Note that the current navigation utilizes unitrees innate controller and it does not currently perform terrain adaptibility. For that low-level-mode is required which is under development.

- Launch High-level driver (LAN)

```
sudo su
source catkin_ws/devel/setup.bash
roslaunch go1_bringup bringup.launch
```

- Launch teleop

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Low-level Mode

- Launch Low-level driver (LAN)

```
sudo su
source catkin_ws/devel/setup.bash
roslaunch go1_base base.launch working_mode:=low_level target_ip:=192.168.123.10
target_port:=8007 local_port:=8090
```

Quick-Start Autonomous Navigation

A pre-configured GO1 can start its navigation via the following commands:

Attention: Note that the current navigation utilizes unitrees innate controller and it does not currently perform terrain adaptibility. For that low-level-mode is required which is under development.

1. Launch High-level driver (LAN)

```
sudo su
source catkin_ws/devel/setup.bash
roslaunch go1_base base.launch working_mode:=high_level target_ip:=192.168.123.161
target_port:=8082 local_port:=8090
```

Note: Odom is published by the driver but more sources of odom can be fused such as visual odometry which we generally provide with the Zed2i camera. Incase you need to perform odometry calculation with another camera VINS_Fusion is a good software to use.

1. Launch navigation driver

```
roslaunch go1_navigation go1_navigation.launch
```

Now it is possible to give the robot navigation goals via movebase. This works only on high-level mode.

1. Launch visualizer

```
roslaunch go1_viz view_robot.launch
```

You can now add point clouds and the raw_rgb_image to view the streamed data.

Network-Setup

- To create a static connection in your own PC, in Ubuntu go to Settings → Network then click on + icon and create a new connection. The first task is to go to IPv4 and change the connection to Manual . The second task is to put the Address IP as 192.168.123.51 and the Netmask as 24 .
- Click save and restart your network. Next is to connect the LAN cable to the robot.
- After a successful connection, the host's local IP can be checked by:

```
ifconfig
```

This should show the host IP which was assigned in the above step. Now its time to check if we can ping the robot or not, to do so type in your host pc

```
ping 192.168.123.161
```

After a successful ping, you can access the robot.

Network-Connection

The GO1 has multiple networks and vary slightly, the following table shows the networks

Device	Network Address	Password
GO1-pi	192.168.123.161	123
GO1-pi (WiFi)	192.168.12.1	123
GO1-nvidia (Head)	192.168.123.13	123
GO1-nvidia (Body)	192.168.123.14	123
GO1-nvidia (Body)	192.168.123.15	123
GO1-MCU	192.168.123.10	N/A

Hotspot

GO1 has its own native hot-spot to which users can connect as well as an Ethernet port to access its on-board computers. Moreover, GO1 has the network address range of 192.168.123.* .

The SSID of the Wifi network of GO1's hot-spot begins with UnitreeRoboticsGO1-000 , where the 000 represent the GO1's model number and the default password is 00000000 or 8 zeros.

Once connected to the WiFi network, one can access GO1's main pc ip address 192.168.12.1 . To achieve this enter the command terminal of the computer that has connected to the WiFi hot-spot of the robot and enter the following:

- To connect with the on-board PC (WiFi):

```
ssh -X pi@192.168.12.1
123
```

- To connect with the on-board PC (LAN):

```
ssh -X pi@192.168.12.1
123
```

- To connect with the nvidia1 (LAN):

```
ssh -X unitree@192.168.123.13
123
```

- To connect with the nvidia2 (LAN):

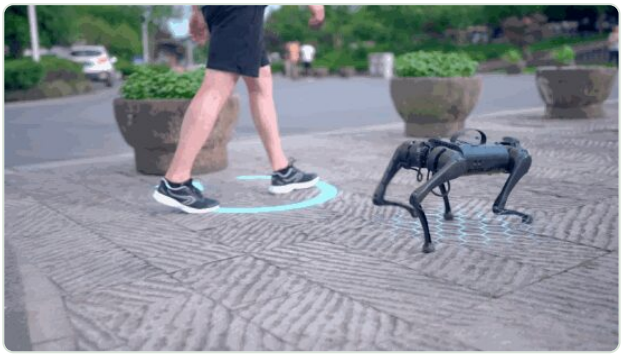
```
ssh -X unitree@192.168.123.14
123
```

- To connect with the nvidia3 (LAN):

```
ssh -X unitree@192.168.123.15
123
```

5 GO1 Operation – Manuals, Calibration & Firmware

GO1 Manuals



Go1 Follow

Important: The following text are hyperlinks to demonstratory videos. Click on them to get redirected to the websites. GO1 Quick Guide GO1 User Manual GO1 Ultrasound Manual GO1 Camera SDK Manual GO1 Unitree SDK Tutorial GO1 Unitree Github GO1 Unitree SDK Github GO1 Unitree Docs (Depreciated) Manufacturers Documentation (Requires Translation Plugin)

Attention: It is highly recommended to view the quick start manual and user manual before using the robot to ensure safe and correct operation.

Note: Currently, we do not have the ultrasound SDK's but will be provided as soon as available.

GO1 Apps

- GO1 iOS
- GO1 Android

GO1 Network

Once the apps are installed users can connect to the native hot-spot of GO1 or through the Ethernet port to access its on-board computers. GO1 has the network address range of 192.168.123.* . The SSID of the Wifi network of GO1's hot-spot begins with UnitreeRoboticsGO1-### , where the hash tags represent the GO1's model number and the default password is 00000000 or 8 zeros.

Once connected to the WiFi network, one can access GO1's app, web-control, and PCs.

Important: It is recommended to connect to the GO1's LAN when developing and testing for better connectivity.

GO1 Basic Operations

A video demonstration on starting-up the robot and performing various actions is given below. It also includes instructions on the activation of companion mode.

- GO1 Remote Control Operation

Good to know commands:

- Walking: START (Press double times)
- Running: L2+START
- Standing: START
- Jump in direction: R2+(direction key)
- Undamped state: L2+B
- Prone position: L2+A (Press double times to go onto standing position)

In standing position:

- Rollover: L2+Y
- Bow with Hands: L1+A (Press START to cancel)
- Dance type 1: L1+X (Press START to cancel)
- Dance type 2: R1+X (Press START to cancel)

In advanced mode (Triggered by pressing START two times rapidly):

- Obstacle avoidance: Y (On)
- Obstacle avoidance: X (Off)
- Stair climbing: A (On)
- Stair climbing: B (Off)

> **Note:** Some features are bound to the version of the GO1.

- GO1 Web Control

Once connected to the WiFi of the GO1. You can access the GO1 website by entering the following in the web addess bar 192.168.12.1 . The GO1's Firmware version, motor temperature, vision, and control can all be accessed in here.

- GO1 Unitree Gazebo

GO1 Calibration

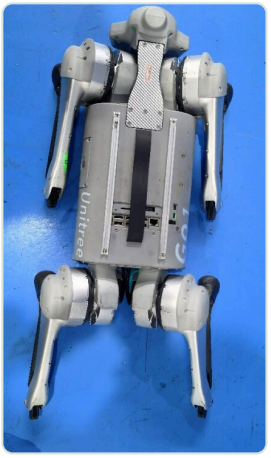
Robot Drift

1. Turn on the robot and the remote.Once the robot is fully booted and stands up, press start

2. While the robot is trotting press the left or right arrow keys opposite of the Go1 drift on the controller keypad
3. Continue pressing the keys until drift stops and press b to save (this has to be done before the robot trotting stops)

IMU Calibration

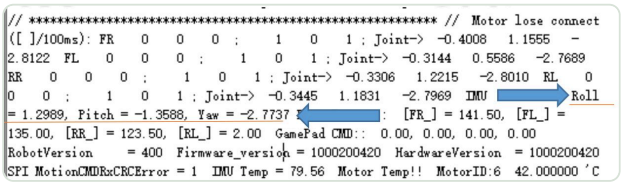
1. If the robot is standing, press L2 + A twice to make the robot lie down.
2. Press L1 + L2 + Start multiple times until the robot enters non-damping mode (you should be able to move the leg joints freely without resistance).
3. Press L1 + B and position the legs so they are close to the ground (as shown in the image below).



IMU Callibration

1. Connect the robot via the Type-C port and use the XCOM software to read the data via the serial port. Focus on the Roll and Pitch values in the software. Slowly move the body until the values approach 0, or use a flat surface for this step.

IMU Calibration Port



IMU Calibration Roll Pitch

1. Press L2 + B and let the robot maintain this position for 3 minutes.
2. Restart the robot.

This process helps correct fuselage skew issues, ensuring the robot's body and ground remain parallel.

Leg Calibration

Some Go1 have different firmware. For most of the Go1's follow until step 2, and then continue from the calibration video.

1. Turn on the robot dog, L2+B makes the robot dog lie down.
2. Press L2+B twice to make the robot dog enter the undamped mode

Important: Twisting the leg joints of the robot dog will not feel damping at this time.

Follow the video guide for the rest of the video guide , as the procedure is the same.

Remote Calibration

- Hold the remote control but do not touch the joystick.
- Press the remote control buttons F1 and F3 and release them at the same time.

At this time, the remote control will emit a continuous drip ~ drip ~ sound (1 time / second) to indicate that it has entered the calibration mode.

- After entering the calibration mode, move the left and right joysticks to full rudder and rotate it several times until the drip ~ drip ~ sound stops, and the calibration is ready.
- Press F3 once to make the calibration take effect and complete the calibration.

Note: Please do not touch the joystick before calibrating, only enter the calibration mode to move the joystick. After calibration, you can view the status of the joystick after calibration through APP. (reference Page: 33 of the user manual)

Robot Falling Over

- Replace the robots foot paws. (In the video tutorials)

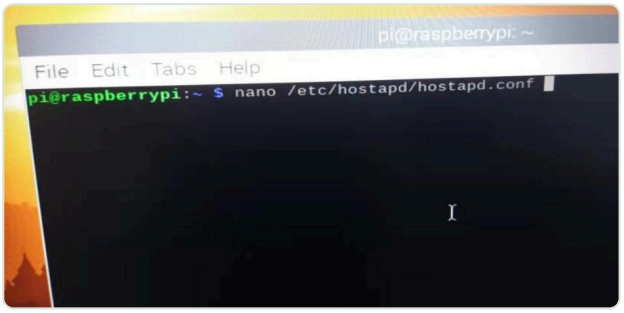
GO1 (EDU) Hotspot Password

The default GO1 Hotspot password is 00000000 .

- To change the GO1 password connect to the GO1 via Ethernet.

Go1 Connection

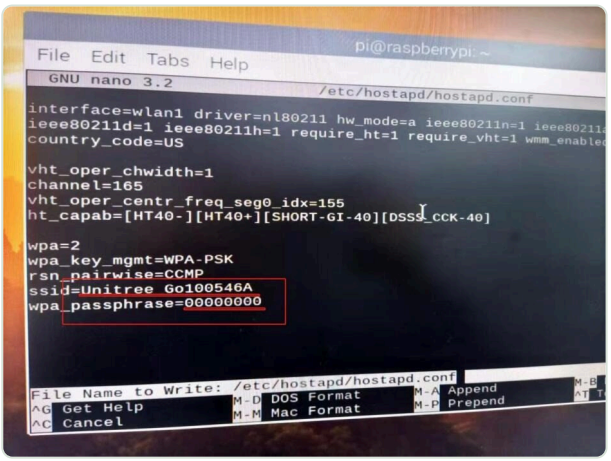
- SSH into the pi as described in Network-Setup .



Go1 Ssh

- Edit the hotspot connection via the nano command.

```
sudo nano /etc/hostapd/hostapd.conf
```



Go1 Hotspot

- Reboot the PC

```
sudo reboot
```

GO1 External Power-Supply

Important: The procedure described below is only for development purposes. It turns on the robot without a battery inside from the external power supply. The power supply should atleast be able to provide 24V/20A .

- GO1 Power-supply

1. Remove the battery from the GO1.
2. Ensure the GO1 is placed on the ground with all four paws as well as knee joints touching the ground. It is advised to place robot on some carpet or similar compressible surface to prevent damage to the joints.
3. Connect the cable to the connection ports on the GO1's back. The GO1 requires 24V with around 20A of current. At startup it may take up to 20A of current. Be vigiliant and make sure the XT30 male pins are in the correct orientation when plugging in.

GO1 Serial Port Check

- Follow the instructions from this manual using this executable .

GO1 Firmware Update

Warning: This procedure may disrupt and potentially render the GO1 unusable. It is recommended to not upgrade if all functionalities are working correctly, GO1 Firmware Update

6 Pre-requisite Software – ROS Installation

Pre-requisite Software

Note: All the ROS related software should be installed/run on remote PC/Rasberry Pi/Nvidia board, nothing needed to be installed on robots control board if bought with the ROS pkg.

[Ubuntu 20.04 - ROS NOETIC]

Ros Noetic

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 or GO1-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 typie 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, what this means is that each time you load up your terminal, it will automatically find the built-in ROS packages.

```
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
```

Once the ROS packages are set-up the next step would be to set up a workspace in which you can add pre-built packages to control your robot.

Catkin Workspace

1. The first step is to ensure that you have correctly setup you ROS noetic. It can be checked by:

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

1. Next, install catkin_tools:

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

1. The next step is to create, your workspace. This is typically created in the home directory following the standard ROS convention, once built, you should see a build and devel folder next to your src folder:

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

1. To enable the workspace, you must type in:

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

1. To avoid typing this everytime you start a terminal, you'll add this into the last line of the bashrc file:

```
gedit ~/.bashrc
```

- Add the following line at the end of the bashrc file:

```
source /home/<your_computer_username>/catkin_ws/devel/setup.bash
```

- Everytime you do a catkin build , it is necessary to either restart all your terminals, or source them.
- To verify that everything is working correctly, type the following into the terminal:

```
echo $ROS_PACKAGE_PATH
```

[Ubuntu 18.04 - ROS MELODIC]

Melodic

The ROS driver for the robots is initially supported for the ROS Melodic . All of ROS related software should be installed in a remote PC/Rasberry Pi/Nvidia board that is onboard the robot. The robot ships either with Rasberry Pi or Nvidia board as an upgradation kit. The steps for 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 robots control board.

PC Setup

ROS Melodic installation requires Ubutnu 18.04 Bionic, Artful and supports macOS, Windows upto varying extent. But for this installation Ubuntu 18.04.x Bionic is recommended.

1. Ubuntu 18.04 can be downloaded from here .
2. The installation instruction for Ubuntu can be found here

ROS-Melodic Installation

The ros driver provided can be run either on a remote PC or on board robot computer. Therefore, ROS only needs to be installed on any one of them. Usually, the on board computer have already pre-installed ROS distribution, so below one will only be applicable to a remote computer. Run following commands to install ROS Melodic or you can simply follow the instructions .

1. Enter the Ubuntu terminal and typie 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-melodic:

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

1. Finally, you will add ROS environment to your bash file, what this means is that each time you load up your terminal, it will automatically find the built-in ROS packages.

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

Once the ROS packages are set-up the next step would be to set up a workspace in which you can add pre-built packages to control your robot.

Catkin Workspace

1. The first step is to ensure that you have correctly setup you ROS melodic. It can be checked by:

```
cd ..
source /opt/ros/melodic/setup.bash
```

1. Next, install catkin_tools:

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

1. The next step is to create, your workspace. This is typically created in the home directory following the standard ROS convention, once built, you should see a build and devel folder next to your src folder:

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

1. To enable the workspace, you must type in:

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

1. To avoid typing this everytime you start a terminal, you'll add this into the last line of the bashrc file:

```
gedit ~/.bashrc
```

- Add the following line at the end of the bashrc file:

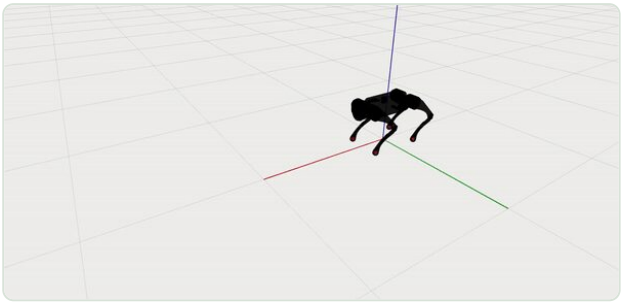
```
source devel/setup.bash
```

- Everytime you do a Catkin build it is necessary to either restart all your terminals, or source them.
- To verify that everything is working correctly, type the following into the terminal:

```
echo $ROS_PACKAGE_PATH
```

7 GO1 Software Setup & Build

GO1 Software Setup



Go1 Rviz

Requirements

This driver requires a system setup with ROS. It is recommended to use Ubuntu 18.04/Ubuntu 20.04 with ROS melodic/noetic.

To make sure that robot control isn't affected by system latencies, it is highly recommended to connect the robot via lan. The driver should run intially on remote PC but can also be run on Rasberry Pi/Nvidia board on robot.

Building

Please follow the pre-requisite section.

- ROS Melodic
- ROS Noetic
- GO1 package Email <support@mybotshop.de>

Optional

- Gazebo9 (melodic)
- Gazebo11 (noetic)

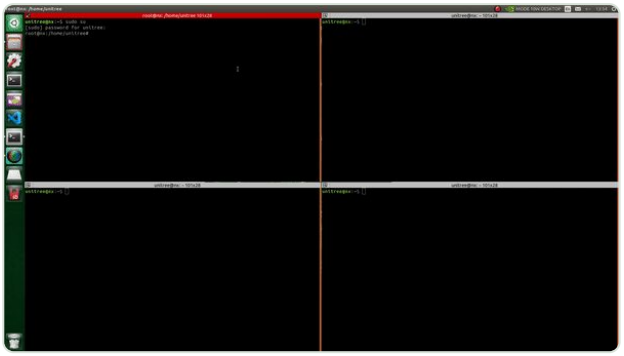
Software Overview

Quadruped robots work in two modes:

1. High level mode
2. Low level mode

User can be in one of these modes and they can not be switched after running.

High level mode



A1 Launch

High level mode is mainly for walking and running. This mode too has two modes normal and motion mode which can be distinguished from their ips (For more details see Quadruped software guide). At the moment this driver is working with default configuration.

In high level mode the robot can walk/run. To run the robot in high level mode run the following command as root user (after turning on the robot and connecting through lan, the robot will be standing):

```
sudo su
source ~/catkin_ws/devel/setup.bash
roslaunch go1_base base.launch
```

Note: Note that in the ”base.launch” file the target ip needs to be changed to 192.168.123.161 when connect to Go1 via LAN where as ip will be 192.168.12.1 when connected via WLAN. Similarly, target port should be 8082 and working mode will be high_level.

With base.launch launches a communication channel between the robot and remote PC. Then later in /base_node/cmd_vel a ROS node is run.

```
Node [high_level_driver]

Publications: state [go1_legged_msgs/HighState]

Subscriptions: cmd_vel [geometry_msgs/Twist]

Services: set_body_pose [go1_legged_msgs/SetMode]
```

The robot pose can be set using the robot set_body_pose service. Robot state is published in state topic. This message contains a lot of information. For more information see high state message .

Using the driver

The node subscribes to base_cmd_vel topic. Here special attention needs to be paid. As the robot is holonimic so different Twist message fields determine different movement. A nice tool here would be teleop_twist_keyboard . Here in the teleop holonomic and non holonomic modes can be used(Be aware key strokes and their usage).

Low level mode

In this mode all the motors can be controlled directly. For that one needs to first change the operation mode either to Servo mode or Electronic brake mode.

When using the lowlevel it is necessary to switch the robot to normal mode:. When the robot is turned on and stands up on its own:

1. L2+B (Together) robot will get down
2. L1+L2+START (Together)

At this point you can run the routine

> **Important:** The robot will fall over when running the low-level routine, please suspend the robot before using it!

Low level mode has again three control modes:

1. Position
2. Velocity
3. Torque

In low level mode joint level control can be achieved. In Quadruped Go1 this low level mode can be in three different levels i.e. Position, Velocity, Torque. All three of them can be used with this driver by publishing appropriate messages. Unlike high level mode robot communication is achieved inside the driver node so no seperate nodeshould run. After turning on the robot and connecting through lan run the followingcommand:

```
sudo su
source ~/catkin_ws/devel/setup.bash
roslaunch go1_base base.launch
```

> **Note:** The base.launch file the target ip needs to be changed to 192.168.123.10 when connect to Go1 via LAN. Similarly, target port should be 8007 and working mode will be high_level.

A node with follwoing information will be run:

```
Node [low_level_driver]

Publications: state [unitree_legged_msgs/LowState]
              joint_states [sensor_msgs/JointState]

Subscriptions: joint_cmd [qre_msgs/JointCMD]

Services: set_body_pose [qre_msgs/SetBodyPose] -> Not implemented yet
          set_control [qre_msgs/SetControl]
```

The node subscribes to joint_cmd topic. Again here special attention to be paid as well for the joints. For joint sequence see qre_msgs README. The driver publishes two topics state topic pubslihes the information from the robot. For more details see low state .. message . Joint state messages are also published. set_control service is can change the robot three modes namely i.e. Position, Velocity, Torque(Case sensitive). For every control level here similar joint messages need to be filled in joint_cmd topic other fields are not considered.

```
Position -> JointCMD.q
            JointCMD.Kd
            JointCMD.Kd
Velocity -> JointCMD.dq
            JointCMD.Kp
            JointCMD.Kd
Torque   -> JointCMD.tau
```

8 GO1 Software Packages – Reference

GO1 Base

This ros package contains the hardware drivers that communicate with motors of the GO1. This driver can be run on a remote PC.

- For WLAN, High-level mode.

```
roslaunch go1_base base.launch working_mode:=high_level target_ip:=192.168.12.1
target_port:=8082 local_port:=8090
```

- For LAN, High-level mode.

```
roslaunch go1_base base.launch working_mode:=high_level target_ip:=192.168.123.161
target_port:=8082 local_port:=8090
```

- For LAN, Low-level mode.

```
roslaunch go1_base base.launch working_mode:=low_level target_ip:=192.168.123.10
target_port:=8007 local_port:=8090
```

GO1 Camera

This ros package contains the camera drivers that communicate with the GO1 cameras.

```
roslaunch go1_camera camera_13.launch
roslaunch go1_camera camera_14.launch
roslaunch go1_camera camera_15.launch
```

GO1 Control

This ros package enables teleop for the Logitech controller.

```
roslaunch go1_control teleop.launch
```

GO1 Description

This is a ros package that contains the 3D models of the GO1. Additionally, the seperate 3D models are stitched together into a single entity via the universal robot description format (URDF). It contains the models for auxiliary components such as lidars as well. Additions to the robot's 3D model can be made in the xacro/robot.urdf.xacro . To view the robot with simulated dummy drivers. You can run:

```
roslaunch go1_description bringup.launch
```

> **Note:** It is ideal to run this on your own PC when the robot is not attached just to adjust the position of your auxiliary components.

The description is automatically launched with the base driver to show the perceived position of the legs with the robot.

```
roslaunch go1_description description.launch
```

GO1 Legged Msgs

This ros package contains custom ros messages used by the GO1 Base.

GO1 Viz

This ros package is used for visualization of the robots current state both in simulation and in the real-hardware.

```
roslaunch go1_viz view_robot.launch
```

9 Simulation

Simulation

Go1 Gazebo

Unitree GO1 supports two different development environment Gazebo and Webots, with which a virtual robot in the simulation can be programmed for testing in real world virtual evironments. Gazebo are simulated with a fake node and visualization tool RViz is used. The current implementation for Gazebo is working using the opensource CHAMP drivers.

Gazebo support different sensors like IMUs, lidars, cameras and many more. With these simulators and sensors autonomous navigation, slam and other functionalities can be tested.

In this instruction, Gazebo will be introduced which is most widely used among ROS developers.

Gazebo Tutorials: http://gazebosim.org/tutorials

> **Note:** In progress

10 GO1 Navigation Setup

GO1 Navigation Setup

A1 Autonomous Navigation

A1 autonomously navigating in a warehouse.

Important: This tutorial is applicable for the Go1 in high-level control mode.

Hardware Requirements

```
1. Autonomous ground vehicle (AGV)
2. Camera (Visual Odometry)
3. Inertial measurement unit (IMU)
4. Logitech controller
```

Recommended Hardware

```
- LiDAR
- Depth Camera
```

- Recommended LiDAR: OUSTER
- Recommended Camera: ZED2

Software Requirements

```
1. Ubuntu 18.04/Ubuntu 20.04
2. ROS-melodic/ROS-noetic
3. QRE Navigation Package
```

Warning: The provided QRE navigation package allows for point to point navigation utilizing GPS coordinates or indoor map coordinates. It integrates the MoveBase package which in turn allows for collision avoidance. This package is currently designed for 2D planes with a planned expansion for

3D planes. Caution must be exercised when using the package around people and or in an un-constrained environment as any fault in sensors or misconfiguration may lead to an accident.

Configuration

In the provided package, several configuration have to be adjusted which are:

- Localization GPS localization config file Odom localization config file
- MoveBase Base local planner params Costmap common params Global costmap params Local costmap params Movebase params

In-depth information is documented by the authors of the robot localization package.

GPS Localization Config

Emlid

In this configuration file, we combine three sensor readings, which are the robots odom, imu, and the gps.

- frequency : 20 The frequency depicts the rate at which the node publishes information.
- two_d_mode : true It tells to ignore the height readings from the incoming sensors as we are navigating in 2D.
- publish_tf : true Publishes a transform from the map frame to the odom frame
- transform_time_offset : 2 Offsets the transform as some packages require transforms to be future off-set by a few seconds.

Setting the sequence of transforms for this localization node.

- odom_frame : odom
- base_link_frame : base_link
- world_frame : map
- map_frame : map

It is important to change the odom to the odom published by your robot.

- odom0 : /your_robot/odom This is typically published by the robots controller
- odom0_config : [false, false, false, false, false, false, true, false, false, false, false, false,false, false, false] The boolean values are (X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za). v is velocity and a is acceleration
- odom0_differential : true This indicates whether the odom velocity should be computed from the given x and y positions.
- imu0 : /imu/data
- imu0_config : [false, false, false, false, false, true, false, false, false, false, false, true, true, false, false] The boolean values are (X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za). v is velocity and a is acceleration
- imu0_differential : false
- odom1 : /odometry/gps This is typically published by the navsat node. Ensure that the correct topic is used according to what you have assigned.
- odom1_config : [true, true, false, false, false, false, false, false, false, false, false, false, false, false, false] The boolean values are (X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za). v is velocity and a is acceleration
- odom1_differential : false

It is highly important to tune the process noises.

- process_noise_covariance : This determines on how much to trust the incoming data. Each line represents X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za. Values for these are assigned along the diagonals.
- initial_estimate_covariance : This determines on how much to trust the initial data. Each line represents X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za. Values for these are assigned along the diagonals.

For more information, the following repository by MethylDragon has a great explanatory guide.

Odom Localization Config

In this configuration file, we combine two sensor readings, which are the robots odom, and imu.

- frequency : 20 The frequency depicts the rate at which the node publishes information.
- two_d_mode : true It tells to ignore the height readings from the incoming sensors as we are navigating in 2D.
- publish_tf : true Publishes a transform from the odom frame to the map frame
- transform_time_offset : 1 Offsets the transform as some packages require transforms to be future off-set by a few seconds.

Setting the sequence of transforms for this localization node.

- odom_frame : odom
- base_link_frame : base_link
- world_frame : odom

It is important to change the odom to the odom published by your robot.

- odom0 : /your_robot/odom This is typically published by the robots controller
- odom0_config : [false, false, false, false, false, false, true, false, false, false, false, false,false, false, false] The boolean values are (X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za). v is velocity and a is acceleration
- odom0_differential : false This indicates whether the odom velocity should be computed from the given x and y positions.
- imu0 : /imu/data
- imu0_config : [false, false, false, false, false, true, false, false, false, false, false, true, true, false, false] The boolean values are (X, Y, Z, roll, pitch, yaw, Xv, Yv, Zv, rollv, pitchv, yawv, Xa, Ya, Za). v is velocity and a is acceleration
- imu0_differential : false

MoveBase Local Planner DWA

TEB planner TEB planner tuning

> **Important:** It is important to note that the DWA planner is no longer used and we use the TEB planner . Configuration for the teb planner can be found in the TEB planner wiki. An excellent resource for tuning your TEB planner is vailable in TEB planner tuning .

It is recommended to configure the move base planner to ensure correct movement. Several important ones are:

- meter_scoring : true When true distance is measured in meters.
- yaw_goal_tolerance : 0.157
- Tolerance in radians for movebase.
- xy_goal_tolerance : 0.25
- Tolerance in xy coordinates for movebase.
- path_distance_bias : 0.35
- Weighting factor of how close the robot should stay to the path.
- goal_distance_bias : 1.0
- Weighting factor of how close the robot should attempt to reach the goal.
- heading_lookahead : 0.325
- Tolerance of looking ahead for available space for turning.

For more information, the following MoveBase Wiki has a brief explanation.

Costmap Common Params

The costmap will be used to integrate and utilize your existing sensor for collision avoidance.Important parameters for this are:

- footprint : [[-0.21, -0.165], [-0.21, 0.165], [0.21, 0.165], [0.21, -0.165]]
- The footprint defines a square size for the robot. The robots mid point is taken as 0,0 and the the corner points are the distance from the mid point.
- footprint_padding : 0.1
- Safety-gap that inflates the original footprint.
- obstacle_layer :
- Sensors that are utilized for sensing. Typically, scan is used for LiDARs and the sort. Important is to change the sensor_frame to match the URDF and the topic to match the input message of the LiDAR.

For more information, the following Costmap Wiki has a brief explanation.

Global Costmap Params

The global costmap of movebase node, as the name states publishes the global costmap. It takes the map, provided by the map server as a static layer. An important point to note is that some warning messages may occur if the on board computer system do not compute the transforms fast enough. These warnings can be ignored.

Local Costmap Params

The local costmap of movebase node publishes the local costmap. It ypically requires the odom frame to operate. However, as we want to depend solely on our GPS, we utilize the map frame for the local costmap as well.

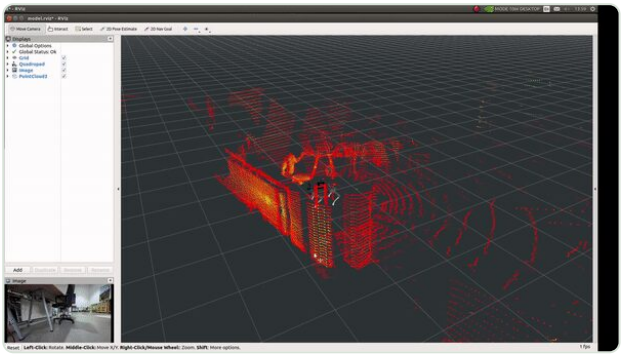
MoveBase Params

The movebase params are important for changing the planner frequency and controller frequency. Usually the default settings are sufficient.

- recovery_behavior_enabled : false
- It rotates the robot to recover from faulty planning failures or positioning, at times this is not desirable and may need to be turned off.

11 GO1 3D SLAM Setup

GO1 SLAM Setup



A1 Point Cloud

Hardware Requirements

1. 3D Lidar
2. Inertial measurement unit(imu)
3. Global positioning device(GPS)(optional)

Software Requirements

Software requirements for packages provided by MYBOTSHOP comes with everything configured (also bash scripts provided in case new installations are needed). We therefore recommend to check/install driver/packages required by third party providers.

3d SLAM Configurations

By default mbs_slam nodes requires /mbs/points(sensor_msgs/PointCloud2) to be published. All other sensor informations are optional. Indepth information on the configuration parameters can be found in koide3

GPS

enable_gps: True in case gps readings are provided. The mbs_slam node needs typically supports 3 types of GPS messages:

- /mbs/geopoint (geographic_msgs/GeoPointStamped)
- /mbs/navsat (sensor_msgs/NavSatFix)
- /mbs/nmea_sentence (nmea_msgs/Sentence)

From all of the above mentioned topics only longitude, latitude, and altitude are used and rest of the fields in the messages are ignored.

IMU Acceleration

enable_imu_acc : By default acceleration resulting from sensor motion is ignored therefore it is useful to provide this paramter (Do not set bigger values for this constraint.)

IMU Orientation

enable_imu_ori : In case the provided IMU has a reliable magentic sensor, orientation can be added as a 3d orientation constraint. In case of external magentic disturbances this paramter should be set to false.

Floor detection

For largescale flat indoor environments, this constraints can be specified. It will reduce the effect of accumulated rotation error.

3D-Localization

The node first does sensor localization using the onboard imu on the lidar. Odomerty prediction based on external imu is optional, if not set constant velocity model is used internally.

qre_localization provides 3d, real-time localization.

ROS Topics

- /odom (nav_msgs/Odometry) Estimated sensor pose in the map frame
- /aligned_points Input point cloud aligned with the map
- /status (hdl_localization/ScanMatchingStatus) Scan matching result information (e.g., convergence, matching error, and inlier fraction)

12 Augmentations & Upgrades

Augmentations

Quadruped GO1 can be upgraded with different sensros and some other accessories. These upgradations require some modifications in the robot software which is detailed below.

Ouster Installation

Ouster

For the ouster attachment, the following package has to be installed in the current workspace. The following tutorial can be followed or the provided description.

Note: This tutorial is to only be followed if the ouster software is not already pre-installed.

1. Install dependencies:

```
sudo apt install -y build-essential libeigen3-dev libjsoncpp-dev  cmake
```

```
sudo apt install -y ros-$ROS_DISTRO-pcl-ros ros-$ROS_DISTRO-rviz ros-$ROS_DISTRO-tf2-
geometry-msgs
```

1. Create a new folder in your catkin workspace src folder named third_party :

```
cd ~/catkin_ws/src
mkdir third_party
cd third_party
```

1. Clone the repository into utils :

```
git clone --recurse-submodules https://github.com/ouster-lidar/ouster-ros.git
```

1. Create your workspace afterwhich you can start configuring the Ouster:

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

1. Connect the ouster with an ethernet cable, and follow the instructions provided in the software user manual chapter 2. It may take ~5 mins for it to be detected.
2. Next navigate to the utils (if the ouster pkg is provided by us) and in there open a terminal and run:

```
chmod +x set_static_ip.py
python3 set_static_ip.py
```

1. Then make open up your ethernet connection that is in the settings and add a new connection. Give your desired name and then navigate to the IPv4 tab. In it change the connection to Manual . In the Addressess , give the address as 192.168.123.2 and the net mask as 255.255.255.0 , and save. Then connect to this ip.

- Verify that the computer is receiving data from ouster via: catkin build source devel/setup.bash

1. Add a new launch file called ouster.launch to the ouster-ros package in the launch directory, and configure the Lidar to your IP and your specifications:

```
<?xml version="1.0"?>
<launch>

<arg name="ouster_ns" default="ouster"/>
<arg name="sensor_hostname" default="192.168.123.35"/>
<arg name="udp_dest" default="192.168.123.6"/>
<arg name="lidar_port" default="46481"/>
<arg name="imu_port" default="45352"/>
<arg name="udp_profile_lidar" default=""/>
<arg name="lidar_mode" default="512x10" />
<arg name="timestamp_mode" default="TIME_FROM_ROS_TIME"/>
<arg name="metadata" default=""/>
<arg name="viz" default="false"/>
<arg name="rviz_config" default="$(find ouster_ros)/config/viz.rviz"/>
<arg name="tf_prefix" default=""/>
<include file="$(find ouster_ros)/launch/sensor.launch" pass_all_args="true"/>

<node pkg="tf" type="static_transform_publisher" name="lidar_to_robot" args="0 0 0.55 0 0 0
/base /os_sensor 100" />

</launch>
```

1. Verify installation via:

```
roslaunch ouster_ros ouster.launch viz:=true
```

ZED2 Installation

Zed 2 Front

This procedure requires that the GO1 has access to internet for initial installation if not done already.

Requirements

- Zed Camera
- Internet connection
- Access to the Nvidia onboard GO1 (The connectors next to the GO1's head)

Procedure

1. Verify CUDA 10.2 is installed in the Nvidia Xavier Tegra board. nvcc -V
2. If not installed follow:

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin

sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600

wget https://developer.download.nvidia.com/compute/cuda/11.6.0/local_installers/cuda-repo-ubuntu1804-11-6-local_11.6.0-510.39.01-1_amd64.deb

sudo dpkg -i cuda-repo-ubuntu1804-11-6-local_11.6.0-510.39.01-1_amd64.deb

sudo apt-key add /var/cuda-repo-ubuntu1804-11-6-local/7fa2af80.pub

sudo apt-get update

sudo apt-get -y install cuda
```

1. Install from the provided QRE package, alternatively you may get the latest version and install from ZED SDK Make sure the Ubuntu version is the 18.04, the CUDA is 10.2, and the Nvidia Xavier SDK is selected for your download
2. Build the provided ZED ROS packages. catkin build
3. Verify installation by running roslaunch zed_wrapper zed2.launch roslaunch zed_display_rviz display_zed2.launch

Python3 (ROS-Melodic)

The buildfarm's binary packages are designed for Python 2 and ROS Melodic. Python 3 is not supported by Melodic and is not recommended. ROS Noetic Ninjemys is the first ROS 1 version with formal support for Python 3. However, if unavoidable, there are a few ways to use Python 3 with ROS Melodic.

Option:1 Source Build

The first method to run python3 is to build ROS-Melodic from source and configuring for Python3, however, one loses the ability to use melodic's build farm to automatically install packages.

> **Note:** A guide is available on python3_source_build .

Option:2 Virtual Environment

Option 2 is to use catkin's virtual environment to run ros-melodic with python3

> **Note:** The package and readme is available on ' roswiki catkin_virtualenv '.

Option:3 Partial Build

Option 3 is to use partially build ros-melodic and then integrate it with python3

> **Note:** A blog explains how to do this python3_partial_build .

Option:4 Custom ROSPY

Option 4 is to use a customized rospy that works with both python2 and python3

> **Note:** The package for it is here rospypi_simple .

13 GO1 OM – Open Manipulator & LiDAR Variant

GO1 OM Quick Start

Go1 Om Ortho

Attention: Please carefully review this document. In case of any uncertainty, it is crucial to seek guidance from specialists, experts, or manufacturers of the assemblies used. The robot should not be put into operation until clarification is obtained. If there are doubts, please refer to the guides below or reach out to a specialist, expert, or manufacturer of the assemblies used. It is imperative not to operate the robot without proper clarification. Ensure that individuals unfamiliar with robotics or these instructions do not operate the robot, as robots can be hazardous in the hands of untrained users.

Note: The robot provided by QUADRUPED Robotics GmbH is designated as a research and development (R&D) device and does not carry a CE Marking or Certificate of Incorporation. A foundational understanding of ROS (Robot Operating System) is essential for users. If unfamiliar with ROS, we recommend consulting the ROS Wiki for guidance. It is important to note that the mentioned robot is classified as a partly completed machine under the Machinery Directive 2006/42/EC and does not bear a CE marking. Clients acknowledge and agree that any information or materials provided by QUADRUPED Robotics GmbH are intended solely for R&D purposes. Services are provided on an “AS IS” basis, without any representation or warranty, whether expressed or implied, including but not limited to merchantability, fitness for a particular purpose, non-infringement, or any other warranty. QUADRUPED Robotics GmbH shall not be held liable for damages, including but not limited to direct, indirect, special, incidental, or consequential damages, arising from the use or inability to use the provided information or materials. This limitation on liability applies regardless of the form of action, be it in contract, tort, or otherwise.

GO1 Robot Interface

Guidelines for interacting with the robot, relevant to GO1’s built-in computers

1. Static Network Connection

For initial setup, use a LAN cable to configure the robot's WLAN network.

1. On your PC (not the robot), go to Settings → Network in Ubuntu, click on + to create a new connection.
2. Change the connection to Manual in IPv4 settings,
3. set the IP address to 192.168.123.51, Netmask to 24, save, and restart your network.

Once connected successfully, check the host's local IP by typing in the Host PC's terminal.

After establishing a successful connection, verify the host's local IP using the Host PC's terminal with the command:

```
ifconfig
```

Next, ping the robot:

```
ping 192.168.123.14
```

Access the robot via SSH:

```
ssh -X unitree@192.168.123.14
```

The default password is: 123

> **Note:** At times, the presence of additional networks can create interruptions when connecting to the GO1. It is advisable to have only the connection to the robot active, while deactivating all others.

2. WiFi Connection

Connect to the GO1 WiFi using its hotspot with the default password '00000000'.

Access the robot via SSH:

```
ssh -X pi@192.168.12.1
```

The default password is:

```
123
```

Then you can ssh again into the Nvidia through the pi

```
ssh -X unitree@192.168.123.14
```

The default password is:

```
123
```

Note: The LAN connection is faster and ideal for development.

3. Screen Connection

Another method for connecting to the robot involves using an HDMI cable along with a mouse and keyboard. This setup enables connecting the robot to your local network, allowing subsequent connections over WiFi.

4. Enabling Internet

To enable internet connectivity on the GO1, ensure it is connected via an LAN cable with internet access, and then simply run:

```
sudo ip link set eth0 down && sudo ip link set eth0 up
sudo dhclient eth0
sudo apt update
```

Note: It may take some time for it to get detected, you can check by running ping google.com .

By default, internet is not required

Initialization

1. Powering on the GO1

- Press the battery button once and then press the battery button again, holding it for 3 seconds.
- Press the remote button located on the bottom side of the controller once, and then press it again, holding it for 3 seconds (similar to the battery).

2. Powering the Open Manipulator

Once the GO1 has completed its boot process, proceed to power on the Open Manipulator. This can be done either by connecting the power connector on the top or by activating the power button situated on the U2D2 board within the top housing.

Note: Avoid turning on the Open Manipulator while the GO1 is still in the booting process, as doing so alters its internal mode and may impede communication with the GO1 SDK.

Installation

Please Clone the repository from out github accout to your Ros1 workspace by following the command:

```
git clone https://github.com/MYBOTSHOP/qre_go1.git
```

Once the repository is downloaded, navigate to your ROS workspace and install the ROS dependencies.

```
rosdep install --from-paths src --ignore-src -r -y
```

Grant permissions and execute to install GO1 Navigation dependencies.

```
sudo chmod + x go1_install.bash && ./ go1_install.bash
```

Now build and source your workspace

```
catkin build -DCMAKE_BUILD_TYPE=Release
source devel /setup.bash
```

Note: For arm64 (GO1 Nvidia), sometimes the build is not detected so you can replace these lines in CMakeLists.txt in go1 base:

```
if ("${CMAKE_SYSTEM_PROCESSOR}" MATCHES "x86_64.*")
    set (SYSTEM_ARCHITECTURE amd64)
endif ()
if ("${CMAKE_SYSTEM_PROCESSOR}" MATCHES "aarch64.*")
    set (SYSTEM_ARCHITECTURE arm64)
endif ()
```

followed by

```
set (SYSTEM_ARCHITECTURE arm64)
```

GO1 Operation

If you've connected through LAN, you can initiate and operate the robot using

```
sudo su
source <your_workspace>/devel/setup.bash
roslaunch go1_bringup bringup.launch
```

1. Teleoperation

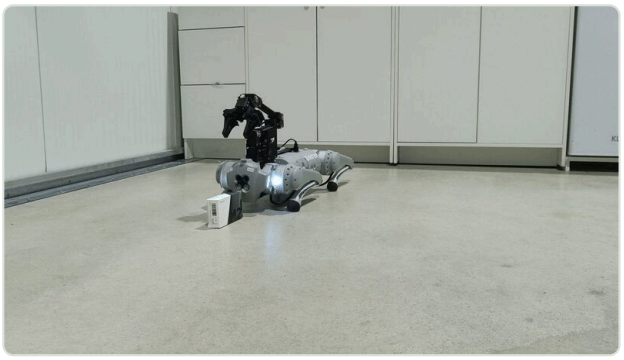
```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

2. Visualization

```
roslaunch go1_viz view_robot.launch
```

Note: This can be run on your own local PC or on the GO1, both should work. Ensure that if you are connecting via WiFi the go1 base launch file has the target ip set to WiFi or else to LAN.

GO1 & Open Manipulator Operation



Pick Place

Note: These are found in the same GO1 repository under the github branch name melodic-robotis.

To launch the GO1 with the Open Manipulator, use the following command:

```
roslaunch go1_bringup mmp_bringup.launch
```

For the initial setup of the GO1 with the Open Manipulator, grant permissions to the port using the following command:

```
sudo chmod a+rw /dev/ttyUSB0
```

1. GO1 Open Manipulator Namespace

Take note that when using the GO1 MMP (Mobile Manipulation Platform) launch file, all GO1 packages are contained within the namespace “GO1” to avoid conflicts with the open manipulator’s packages. Utilize ROS commands to navigate and understand these namespaces.

```
rosnode list
rostopic list
rosservice list
```

By default, the GO1’s normal bring-up does not contain namespaces for every node.

Attention: Ensure to execute this on the GO1 PC with the connected open manipulator! The default PC for this purpose is the Nvidia board with the IP address 192.168.123.14.

2. Open Manipulator Gripper

Open manipulator can be controller via the Moveit arm controller group. The gripper can be controller via the services

To open the gripper

```
rosservice call /go1_robotis/goal_tool_control "planning_group: 'gripper'
joint_position :
joint_name :
- ' gripper '
position :
- 0.01
max_accelerations_scaling _factor:0.0
max_velocity_scaling_factor:0.0
path_time : 0.0 "
```

To close the gripper

```
rosservice call /go1_robotis/goal_tool_control "planning_group : 'gripper'
joint_position :
joint_name :
- ' gripper '
position :
- -0.01
max_accelerations_scaling _factor:0.0
max_velocity_scaling_factor:0.0
path_time : 0.0 "
```

For different positions, you may use any value between -0.01 to 0.01, where 0.01 is for opening and -0.01 is for closing.

3. Open Manipulator Demo

Combo

The GO1 demo package can be employed to evaluate the Moveit package's functionality. Initially, make sure the mmp_bringup package is launched:

```
roslaunch go1_bringup mmp_bringup.launch
```

Subsequently, execute the GO1 demo package:

```
rosrun go1_demo demo.py
```

You can input commands to navigate to pre-defined poses like home, stand, left, right, etc.

4. Open Manipulator Powering Off

When shutting down the open manipulator driver, it is advisable to use the default home pose from Moveit to ensure it rests in a suitable standby position. Failure to do so may result in the Open Manipulator falling.

GO1 & Robosense LIDAR Operation

Go1 Rs Out

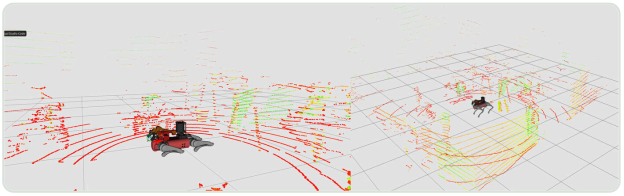
> **Note:** These are found in the same GO1 repository under the github branch name melodic-robotis-robosense.

To launch the GO1 with the Robosense Lidar, use the following command:

```
roslaunch go1_lidar lidars.launch
```

Running this command initiates the lidar driver connected to the robot, and you can observe the point cloud generated by that lidar by executing the visualization command:

```
roslaunch gi1_viz view_moveit.launch
```



Go1 Scan

GO1 Camera

For this, a ROS multi-machine setup is required. Clone this package to the three Nvidia boards on the GO1 with the IP addresses 192.168.123.13, 192.168.123.14, and 192.168.123.15 respectively. When used in conjunction with robot upstart (not covered here yet), it should start up every time the robot boots.

> **Note:** The default camera driver auto-startup on each Nvidia needs to be disabled.

To set up multi-machine functionality, add the client PC details from the code above to the .bashrc file. Replace the IP 192.168.123.— with the Nvidia IP, which can be found using ifconfig (e.g., 13 in this case).

1. Disable Camera Driver startup

Run this on the PC and unlink the camera driver

```
gnome-session-properties
```

In some versions

```
./Unitree/autostart/camerarosnode/cameraRosNode/kill.sh
```

2. GO1 Navigation

Intended to be run on the host PC.

Odometry Navigation

For navigation based purely on the robot's odometry, start by running the bring-up:

```
roslaunch go1_bringup bringup.launch
```

Next, launch the odometry navigation:

```
roslaunch go1_navigation odom_navi.launch
```

3. GO1 Service Examples

This requires the go1 driver to be running, either bringup.launch or mmp_bringup.launch .

GO1 Stand Down Position

```
rosservice call /set_mode "mode: 5
gait_type: 0"
```

With go1 namespace (used in open manipulator):

```
rosservice call /go1/set_mode "mode: 5
gait_type: 0"
```

GO1 Force Stand Position

```
rosservice call /set_mode "mode: 6
gait_type: 0"
```

With go1 namespace (used in open manipulator):

```
rosservice call /go1/set_mode "mode:
6 gait_type: 0"
```

GO1 Return to Walk Mode Control

```
rosservice call /set_mode "mode: 2
gait_type: 1"-----------
```

With go1 namespace (used in open manipulator):

```
rosservice call /go1/set_mode "mode: 2
gait_type: 1"
```

GO1 Gazebo

Go1 Om Gazebo

Launch Simulation

```
roslaunch go1_gazebo gazebo.launch
```

Unpause simulations by pressing the play button. GO1 (simulation) can be teleoperated via:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/go1/cmd_vel
```

Gaits are untuned and can be changed via go1_gazebo/config/gait/gait.yaml .

GO1 Multimachine — Cameras

To use rviz on your pc for faster updates please setup the multimachine.

1. Multi Machine - Host

For all the terminals on your PC (4-5 recommended), export the following environment variables. The IP should correspond to the IP address of your current host obtained from sudo nmtui or ifconfig .

```
export ROS_MASTER_URI=http://192.168.123.1:11311
export ROS_HOSTNAME=192.168.123.1
export ROS_IP=192.168.123.1
```

2. Multimachine - Client

For all the terminals on the robot's PC, export these environment variables. The ROS master should correspond to the Multi Machine - Host, whereas the ROS IP and ROS hostname should correspond to the robot's IP.

Nvidia 192.168.123.13

```
export ROS_MASTER_URI=http://192.168.123.1:11311
export ROS_HOSTNAME=192.168.123.13
export ROS_IP=192.168.123.13
```

Nvidia 192.168.123.14

```
export ROS_MASTER_URI=http://192.168.123.1:11311
export ROS_HOSTNAME=192.168.123.14
export ROS_IP=192.168.123.14
```

Nvidia 192.168.123.15

```
export ROS_MASTER_URI=http://192.168.123.1:11311
export ROS_HOSTNAME=192.168.123.15
export ROS_IP=192.168.123.15
```

Please note that you should ideally have the go1 description and go1_viz ROS packages built on your PC so that you may view the robot over the network.

3. Multimachine Usage

On the robot, launch:

```
roslaunch go1_bringup bringup.launch
```

On the user's PC, launch:

```
roslaunch go1_viz view_robot.launch
```

1. Control OpenManipulator

```
roslaunch go1_gazebo open_manipulator_control.launch
```

Press the timer start to use the GUI buttons. It is also possible to control via the joint_command topics.

GO1 & Open Manipulator Caution

Dynamixel Error:

1. Power cycle the adapter.
2. Run sudo chmod a+rw /dev/ttyUSB0.

Robot Joint States Not Showing Up:

1. Ensure the robot has fully booted up.
2. Connect the LAN cable after the robot has booted, especially if using DHCP.

GO1's Motion Planning Issue with ROS Multimachine:

1. Wait for the robot to fully boot up before connecting the LAN cable, especially if using DHCP.
2. Use ROS Melodic when interfacing with ROS on GO1 (applicable only for GO1 with Manipulator).

Open Manipulator Falling on the Robot:

Execute the home pose from Moveit to position the gripper in a standby position.

GO1 & Open Manipulator Safety Guidelines

When deploying robotic manipulators, prioritizing safety procedures is crucial to mitigate risks and ensure secure operations. The following guidelines outline key safety measures when working with robotic manipulators:

1. Work Area Safety

- Maintain a clean and well-organized work area to ensure proper sensor functioning and precise manipulation, avoiding clutter and inadequate lighting.
- Avoid operating robotic manipulators in hazardous environments with corrosive substances, extreme temperatures, or sharp objects.

- Ensure only authorized personnel are present during manipulator operation to prevent interference and maintain a safe working environment.

2. Electrical Safety

- Ensure the manipulator's power system adheres to electrical safety standards, conducting regular inspections and maintenance.
- Implement safeguards to protect the manipulator from adverse environmental conditions like moisture or extreme temperatures.
- Regularly inspect power cables and connections, promptly replacing damaged components to minimize electrical risks.

3. Manipulation Safety

- Implement collision detection systems to prevent unintended contact with objects, humans, or other equipment during manipulation tasks.
- Define and enforce safety zones around the manipulator's workspace to minimize the risk of unintended interactions.
- Regularly calibrate and test the manipulator's sensors and systems for precise and reliable performance during manipulation tasks.

4. Emergency Response

- Install an emergency stop mechanism to swiftly halt manipulator operation in unforeseen circumstances or emergencies.
- Clearly mark and communicate emergency stop locations within the manipulator's operational area.
- Conduct regular emergency response drills to ensure personnel are familiar with procedures during unexpected situations.

5. Data Security and Privacy

- Implement robust cybersecurity measures to safeguard the manipulator's control systems and data from unauthorized access or manipulation.
- Ensure compliance with privacy regulations when collecting, storing, or transmitting data captured by the manipulator's sensors.

6. Human Interaction Safety

- Integrate sensors and communication systems to detect and respond to the presence of humans in the manipulator's vicinity.
- Clearly communicate the manipulator's operational status and intentions using visual and audible signals to alert nearby individuals.

- Establish protocols for safe human-robot collaboration, particularly in shared workspaces where manipulators are in operation.

7. Residual Risks

Despite the implementation of safety measures, certain residual risks may persist. These include:

- Impairment of sensor functionality.
- Risk of collisions during complex manipulation tasks.
- Cybersecurity vulnerabilities.
- Unintended human interactions due to unforeseen circumstances.

Robotic manipulators are sophisticated technologies that require correct usage to avoid accidents and ensure a secure environment. Learn and adhere to proper procedures diligently, prioritizing both precision and safety.

GO1 Safety Guidelines

When deploying autonomous mobile robots, prioritizing safety procedures is essential to prevent accidents and ensure secure operations. The following guidelines outline key safety measures when working with an autonomous mobile robot:

1. Work Area Safety

- Maintain a clean and well-lit work area. Cluttered or poorly illuminated spaces can impede the proper functioning of sensors and navigation systems.
- Avoid operating autonomous mobile robots in explosive atmospheres, such as areas with flammable liquids, gases, or dust. The robot's components may pose a risk in such environments.
- Keep bystanders and unauthorized personnel at a safe distance during robot operation to prevent interference with autonomous navigation.

2. Electrical Safety

- Ensure the robot's power system adheres to electrical safety standards. Regularly inspect and maintain power components to prevent malfunctions.
- Implement mechanisms to protect the robot from adverse weather conditions, such as rain or wet environments.
- Regularly inspect power cables and connections and replace damaged components promptly to minimize the risk of electrical issues.

3. Navigation Safety

- Implement obstacle detection and avoidance systems to prevent collisions with objects, people, or other robots.

- Define and enforce safety zones within the robot's operational area to minimize the risk of unintended interactions with personnel or other equipment.
- Regularly calibrate and test the robot's navigation sensors and systems to ensure accurate and reliable performance.

4. Emergency Response

- Install an emergency stop mechanism to quickly halt the robot's operation in case of unforeseen circumstances or emergencies.
- Clearly mark and communicate emergency stop locations throughout the robot's operational area.
- Conduct regular emergency response drills to ensure personnel are familiar with procedures for handling unexpected situations.

5. Data Security and Privacy

- Implement robust cybersecurity measures to protect the robot's control systems and data from unauthorized access or manipulation.
- Ensure compliance with privacy regulations when collecting, storing, or transmitting data captured by the robot's sensors.

6. Human Interaction Safety

- Integrate sensors and communication systems to detect and respond to the presence of humans in the robot's vicinity.
- Clearly communicate the robot's operational status and intentions using visual and audible signals to alert nearby individuals.
- Establish protocols for safe human-robot collaboration, especially in shared workspaces.

7. Residual Risks

Despite the implementation of safety measures, certain residual risks may persist. These include:

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

Autonomous Mobile Robots (AMR) are advanced technologies that require correct usage to avoid accidents and ensure a secure environment. Learn and follow the proper procedures diligently; prioritize both quality and safety.

14 Getting Help & RMA

RMA / Support

How to Raise a Return Merchandise Authorization (RMA) Case

At QUADRUPED Robotics, 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 @ quadruped . 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: QUADRUPED Robotics GmbH c / o 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. Do not ship goods to QUADRUPED Robotics GmbH Office in Leverkusen!

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 .

15 Getting Involved

Getting Involved

Contact support @ mybotshop . de for more information.

16 FAQ

Frequently Asked Questions

Wireless Access GO1

The Go1 can be wirelessly accessed by its onboard hotspot. It usually has the name UnitreeRoboticsGO1-## followed by the model number. The default password is 00000000 .

- The main controller ip:

```
ssh -X unitree@192.168.123.161
```

- The Nvidia controller ip:

```
ssh -X unitree@192.168.123.12
```

Important: The password for both are 123 .

The Unitree App can be used to view the robot as well.

- For recording and visualing its data, it is recommended to use record a rosbag transfer it back to your computer and play it:

- Record rosbag:

```
rosbag record -a
```

- Play rosbag:

```
rosbag play -l <file_name.bag>
```

Display not visible GO1

There are two pcs on the robot. The front side of the robot has some ports they are for Rasberry Pi. Here you can connect the HDMI cable and than restart the robot because if the robot is turned on and than you connect an HDMI adapter then it will not show anything. So you have to restart after connecting the HDMI. For Nvidia the same procedure applies.

Note: The Nvidia's moAlstherboard goes to sleep, so if you connect an external mouse or keyboard you can directly wake it up.