MYBOTSHOP

QUADRUPED ROBOTICS

Unitree A1

User Manual

CONTACT

info@quadruped.de

support@quadruped.de

quadruped.de

Contents

1 Attention

Attention: Read this document carefully before operating the robot. In case of doubt, 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 15, Getting Help & RMA.

It is highly recommended to also view the official Unitree A1 user manual and video guides (unitreerobotics.net, linked from the MYBOTSHOP documentation site) before using the robot, to ensure safe and correct operation.

Note: This documentation assumes you are comfortable working with ROS. If you are not familiar with ROS already, we recommend starting with the official ROS tutorials first.

2 Disclaimer

The A1 robot documented here is sold and supported by *MYBOTSHOP / QUADRUPED Robotics* as the German distributor for Unitree Robotics. Basic ROS 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. This manual documents two parallel driver revisions (v2.8, legacy, and v3.2, current) — always confirm which is installed on your unit before following a build or launch procedure, since they are not interchangeable. Low-level joint-torque control and the field-upgrade procedures (LiDAR, camera, motor ID changes) carry real risk of hardware damage if followed incorrectly — when in doubt, contact support before proceeding. For terms, policies and compliance with local laws and regulations, refer to the Unitree Robotics website and the RMA/support process in Chapter 15.

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 A1

A1 Quadruped Tutorials

A1 Robot

This package supplies Sphinx-based tutorial content to assist you with setting up and operating your A1 quadruped robot. The tutorials topics are listed in the left column, and presented in the suggested reading order.

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 A1 in simulation is valuable whether you are in the testing phase with software you intend to ultimately deploy on a real A1, 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.

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.

Important: The following text are hyperlinks to manuals and repositories. Click on them to get redirected to the websites. A1 User Manual A1 Android App A1 iOS App A1 Unitree SDK Github A1 Starting Robot A1 Robot Video Guide A1 Paw Replacement Video Guide A1 App Video Guide Manufacturers Documentation

4 Pre-requisite Software – ROS Setup

Pre-requisite Software

The ROS driver for the robots is initially supported for the ROS Melodic . All of ROS related software should be install either on remote PC/Rasberry Pi/Nvidia board on 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 needed to be installed on robots control board.

[Ubuntu 20.04 - ROS NOETIC]

_images/ros_noetic.png

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 A1-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]

_images/melodic.jpg

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
```

5 Quick Start, Network & Calibration

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 A1 robot is shown. The software is currently under heavy development. At the moment simples ros drivers are available for the robot along with simulation are available in Gazebo.

Quick-Start Teleop

A pre-configured A1 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

```
sudo su
source catkin_ws/devel/setup.bash
roslaunch qre_ros high_level_mode.launch
```

1. Launch teleop

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Quick-Start Autonomous Navigation

A pre-configured A1 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

```
sudo su
source catkin_ws/devel/setup.bash
roslaunch qre_ros high_level_mode.launch
```

1. Launch qre driver

```
roslaunch qre_driver qre_base_driver.launch
```

This launches the robot state publisher, joint state publisher, IMU publisher, and Odom publisher. Note: The odom requires a visual odometry topic to work which we generally provide with the ZED2 camera. Incase you need to perform odometry calculation with another camera VINS_Fusion is a good software to use.

1. Launch ouster driver

```
roslaunch mbs_ouster ouster.launch
```

1. Launch navigation driver

```
roslaunch mbs_navigation odom_navigation.launch
```

Now it is possible to give the robot navigation goals via movebase.

1. Launch visualizer

```
roslaunch qre_viz viz.launch
```

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

Robot Setup

To start-up the robot the instructional video in the overview should be used which details the opening and use of the A1 manual as well as operation.

Network- Setup

The A1 has multiple networks and vary slightly, the following table shows the networks

Device	Network Address	Password
A1-pi	192.168.123.12	123
A1-nvidia	192.168.123.161	123
A1-MCU	192.168.123.10	N/A
Ouster	192.168.123.51	N/A

Hotspot

> **Note:** A1 has its own native hot-spot to which users can connect as well as an Ethernet port to access its on-board computers. Moreover, A1 has the network address range of 192.168.123.* .

The SSID of the Wifi network of A1's hot-spot begins with UnitreeRoboticsA1-000 , where the 000 represent the A1's model number and the default password is 00000000 or 8 zeros.

Once connected to the WiFi network, one can access A1's Nvidia ip address 192.168.131.161 as well as the Raspberry Pi's ip address 192.168.123.12 . 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 Nvidia's on-board PC:

```
ssh -X unitree@192.168.123.161
123
```

- To connect with the Raspberry Pi's on-board PC:

```
ssh -X unitree@192.168.123.12
123
```

Mobile-App

- Install the unitree app: A1 Android App , A1 iOS App
- Connect to the A1's hotspot and select the vision system to see the A1's camera stream

Direct Power-Supply

1. Remove the battery from the A1.
2. Ensure the A1 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 A1's back. The A1 requires 24V with around 3A 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.

> **Warning:** The procedure described below is only for development purposes. It turns on the robot without battery inside from external power supply. The power supply should atleast be able to provide 24V/3A .

Calibration

Robot Drift

1. On the robot and press start
2. While the robot is trotting press the left or right arrow keys opposite of the A1 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. Power on the robot until it stands.
2. Press L2 + B to make the robot lie down.
3. Press L2 + B twice to enter zero-force mode (motors can move freely).
4. Position the legs so that all four are off the ground (As shown in the picture below), with thighs parallel to the ground.

IMU Callibration

1. Press L1 + B , wait for 3 minutes, then power off the robot.
2. Return the legs to the usual startup position, power on the robot, and check the result.

Leg Calibration

Follow the instructions below to calibrate the legs of Robot A1 properly. This procedure ensures that the robot's leg joints are correctly aligned for optimal performance.

> **Note:** Make sure to follow the steps carefully, and maintain a safe distance from the robot during calibration to avoid injury.

Step-by-Step Calibration

1. Power on the Robot : Ensure that all four legs are touching the ground in their neutral position. Power on the robot by pressing the battery power button with one short press followed by one long press.
2. Wait for Boot Up : Once the robot's boot-up process is complete, it will automatically stand up.
3. Power on the Joystick : Power on the joystick by pressing the power button with one short press followed by one long press.
4. Begin Calibration : Follow the button sequence below to perform the calibration process. Press L2 + A (x2). Press L2 + B (x1) – The robot will sit down. Press L2 + R2 (x1) – Ensure that the motors are now free of tension, and the legs can be moved freely by hand.
5. Adjust Knee Joints : Manually bring the knee joints of both legs together, aligning them at the center of the robot's body. Ensure all four points (both knee joints and feet) are properly aligned, with the connection centered.
6. Prepare for Leg Extension : Maintain a safe distance, as the robot's legs will move outward in the next step. Press L2 + R2 (x1) – The robot's legs will widen. At this stage, the hip joints will be

locked, allowing movement only in the front-back direction, not side-to-side.

7. Align the Legs Using the Calibration Tool : Use the calibration tool to align each leg of the robot, slowly guiding them down to the ground.
8. Finalize Leg Position : After aligning all four legs, complete the process by following these button presses: Press L2 + L1 (x2). Press L2 + R1 (x2). Press L1 + R1 + R2 together (x1).
9. Verify the Calibration : Once these steps are completed, you should notice a reduction in motor noise, and the leg joints should move freely. This indicates that the calibration is successful.
10. Reboot the Robot :

- Power cycle the robot by turning it off and on again to finalize the calibration.

Video Guide

For a detailed visual guide, refer to the leg calibration video guide .

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)

Troubleshooting Guide

You can access the full troubleshooting guide here .

- To proceed with the troubleshooting steps, first download document(EXE) #1
- Then, download document(EXE) #2

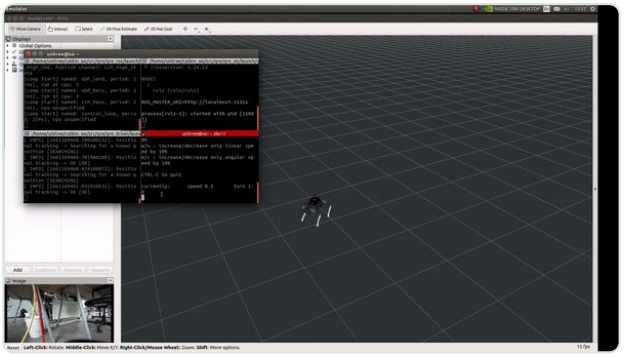
Follow the instructions within the guide and the linked documents for resolving issues with your QUADRUPED A1.

Robot Falling Over

- Replace the robots foot paws.

6 A1 Driver v3.2 (Current)

A1 Driver v3.2



_images/a1_teleop.gif

A1 is intended mostly for academic community. Unitree Robotics has provided an SDK for their robots in C++ and a demo is also provided in ROS . But this ROS driver lacks several ROS features. To cover them this driver is provided.

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
- Gazebo9 (melodic)
- Gazebo11 (noetic)
- Unitree_legged_sdk_3.2

Furthermore unitree_legged_sdk has the following dependencies.

- Boost (version 1.5.4 or higher)
- CMake (version 2.8.3 or higher)
- LCM (version 1.4.0 or higher)

Attention: It is important to check your CPU type via executing lscpu in the terminal. If you have Architecture:x86_64 then nothing is to be changed, if you have Architecture:aarch64 then in the utils/qre_unitree_envs.bash , you must change it to the cpu from amd64 to arm64 . Also in unitree_legged_sdk_3.2, you must change the processor type from amd64 to arm64 .

A script can be found below for automatic installation of all the required dependencies mentioned above and the provided software installation. After downloading and changing directory to the downloaded folder, installation script can be run by the following command:

Note: For the A1 installation package please email support @ mybotshop . de

You first have to give the script root permission via:

```
sudo chmod +x a1_installation_script.bash
```

```
./a1_installation_script.bash
```

Now everything should be installed and built (assuming ROS and Gazebo installed prior to following this tutorial). If there is any issues please contact support@mybotshop.de .

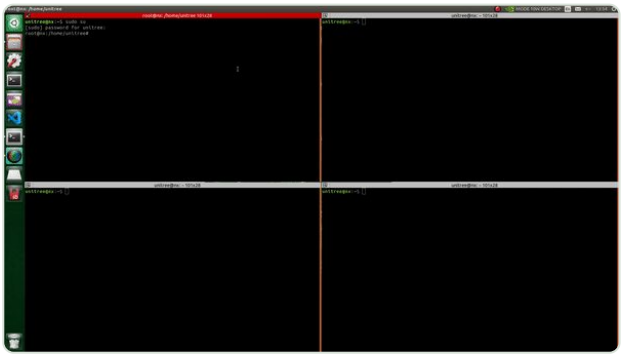
Software Overview

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



_images/a1_launch.gif

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 qre_ros high_level_mode.launch
```

With high_level_mode.launch unitree_legged_real/real.launch launches a communication channel between the robot and remote PC. Then later in qre_unitree_ros/high_level_mode a ROS node is run.

```
Node [high_level_driver]

Publications: state [unitree_legged_msgs/HighState]

Subscriptions: cmd_vel [geometry_msgs/Twist]

Services: set_body_pose [qre_msgs/SetBodyPose]
```

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 message.

Using the driver

The node subscribes to 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).

Body height and walking

Body height and walking demo

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. 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 A1 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 node should run.

After turning on the robot and connecting through lan run the following command:

```
sudo su
source ~/catkin_ws/devel/setup.bash
roslaunch qre_ros low_level_mode.launch
```

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]
```

Position control	Velocity control	Torque control

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
```

Using the driver

For velocity and torque control communication can done easily by just publishing the respected values e.g. by using rqt topic publisher , in terminal:

```
rqt
```

For position control mode the same control can be used through rqt , but to get smoother motions and communication a demo is provided for some predefined joint values. After running the low_level_driver.launch , execute the following command:

```
rosrun qre_ros llm_demo.py
```

PositionControlExample

Low level position control demo with ROS

7 A1 Driver v2.8 (Legacy)

A1 Driver v2.8

A1 is intended mostly for academic community. Unitree Robotics has provided an SDK for their robots in C++ and a demo is also provided in ROS . But this ROS driver lacks several ROS features. To cover them this driver is provided.

Requirements

This driver requires a system setup with ROS. It is recommended to use Ubuntu 18.04 with ROS melodic, however using Ubuntu 16.04 with ROS kinetic should also work.

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

The ROS driver for Quadruped robot is dependent upon following packages:

- ROS Melodic or ROS Kinetic (has not been tested)
- Gazebo8
- unitree_legged_sdk
- aliengo_sdk

Furthermore unitree_legged_sdk and aliengo_sdk have following dependencies.

- Boost (version 1.5.4 or higher)
- CMake (version 2.8.3 or higher)
- LCM (version 1.4.0 or higher)

Attention: It is important to check your CPU type via executing lscpu in the terminal. If you have Architecture:x86_64 then nothing is to be changed, if you have Architecture:aarch64 then in the utils/qre_unitree_envs.bash , you must change it to the cpu from amd64 to arm64 .

A script can be found below for automatic installation of all the required dependencies mentioned above and the provided software installation. After downloading and changing directory to the downloaded folder, installation script can be run by the following command:

Note: A1 installation script download

You first have to give the script root permission via:

```
sudo chmod +x a1_installation_script.bash
```

```
./a1_installation_script.bash
```

Alternatively user can follow the procesure below to install all the dependencies and required software manually.

Warning: Please do not follow the installation guidelienes below for building the software, if you have executed the above mentioned installation script.

So now let's start installing the dependencies one by one.

To install Boost run the following command

```
sudo apt install libboost-all-dev
```

CMake can be installed by the following command:

```
sudo apt install cmake
```

To install LCM , download the latest version here . Unzip the file and change directory to the LCM folder you just downloaded. Follow the steps below:

```
cd ~/lcm
mkdir build
cd build
cmake ../
make
sudo make install
```

To install unitree_legged_sdk :

```
git clone https://github.com/unitreerobotics/unitree_legged_sdk ~/unitree_legged_sdk
cd ~/unitree_legged_sdk
mkdir build
cd build
cmake ../
make
```

To install aliengo_sdk :

```
git clone https://github.com/unitreerobotics/aliengo_sdk ~/aliengo_sdk
cd ~/aliengo_sdk
mkdir build
cd build
```

```
cmake ../
make
```

Before installing ROS driver from Quadruped Robotics, we have to install unitree_legged_real (This will be removed in future versions).

```
source /opt/ros/$ROS_DISTRO/setup.bash
# For ROS melodic Gazebo 9 is supported so here make sure to use your respective Gazebo
version.
source /usr/share/gazebo-9/setup.sh
source ~/catkin_ws/devel/setup.bash
export ROS_PACKAGE_PATH=~/catkin_ws:${ROS_PACKAGE_PATH}
export GAZEBO_PLUGIN_PATH=~/catkin_ws/devel/lib:${GAZEBO_PLUGIN_PATH}
export LD_LIBRARY_PATH=~/catkin_ws/devel/lib:${LD_LIBRARY_PATH}
export UNITREE_LEGGED_SDK_PATH=~/unitree_legged_sdk
export ALIENGO_SDK_PATH=~/aliengo_sdk
# amd64, arm32, arm64
export UNITREE_PLATFORM="amd64"
sudo apt-get install ros-$ROS_DISTRO-controller-interface  ros-$ROS_DISTRO-gazebo-ros-
control ros-$ROS_DISTRO-joint-state-controller ros-$ROS_DISTRO-effort-controllers
ros-$ROS_DISTRO-joint-trajectory-controller
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
git clone https://github.com/unitreerobotics/unitree_ros src/unitree_ros
catkin build
source devel/setup.bash
```

Now everything is installed(assuming ROS and Gazebo installed prior to following this tutorial). The next step is to install ROS driver provided by Quadruped Robotics.

To build the driver download the source code from here and copy/paste it inside catkin_ws/src and run the following commands:

```
source /opt/ros/$ROS_DISTRO/setup.bash
cd ~/catkin_ws && catkin build
source devel/setup.bash
sudo apt update -qq
rosdep update
rosdep install --from-paths src --ignore-src -y
catkin build
source devel/setup.bash
```

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

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 qre_ros high_level_mode.launch
```

With high_level_mode.launch unitree_legged_real/real.launch launches a communication channel between the robot and remote PC. Then later in qre_unitree_ros/high_level_mode a ROS node is run.

```
Node [high_level_driver]

Publications: state [unitree_legged_msgs/HighState]

Subscriptions: cmd_vel [geometry_msgs/Twist]

Services: set_body_pose [qre_msgs/SetBodyPose]
```

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 message.

Using the driver

The node subscribes to 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).

Body height and walking

Body height and walking demo

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. 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 A1 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 node should run.

After turning on the robot and connecting through lan run the following command:

```
sudo su
source ~/catkin_ws/devel/setup.bash
roslaunch qre_ros low_level_mode.launch
```

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]
```

PositionControl

VelocityControlFast

TorqueControl

Position control	Velocity control	Torque control

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
```

Using the driver

For velocity and torque control communication can done easily by just publishing the respected values e.g. by using rqt topic publisher , in terminal:

```
rqt
```

For position control mode the same control can be used through rqt , but to get smoother motions and communication a demo is provided for some predefined joint values. After running the

low_level_driver.launch , execute the following command:

```
rosrun qre_ros llm_demo.py
```

PositionControlExample

Low level position control demo with ROS

8 Perception – RealSense Depth Camera

Perception-Driver

Unitree quadrupeds Aliengo and A1 are equipped with depth cameras. Here for the sake this documentation Unitree A1 is tested which has Intel Realsense D435i . Moreover for this specific model we have Rasberry Pi 4b (Later will be referred to RPI), so all the installation instructions are specific to this specific model.

Driver Installation

Intel RealSense has its ROS driver available. This ROS driver is dependent on librealsense RealSense SDK which also has a few dependencies. So here we'll start by first installing dependencies for RealSense SDK.

- Let's start by updating, upgrading the system and installing dependencies and some useful tools.

```
sudo apt-get update && sudo apt-get dist-upgrade
sudo apt-get install automake libtool vim cmake libusb-1.0-0-dev libx11-dev xorg-dev
libglu1-mesa-dev
```

- Next task is to expand the filesystem in RPI by selecting the menu entry Advanced Options and reboot.

```
sudo raspi-config
```

- After successful reboot now increase SWAP size in /etc/dphys-swapfile by adding CONF_SWAPSIZE=2048 and commenting out the deafult one(usually its CONF_SWAPSIZE=100)

```
sudo vim /etc/dphys-swapfile
```

- Apply the above mentioned SWAP changes by:

```
sudo /etc/init.d/dphys-swapfile restart swapon -s
```

- Create a new udev rule for the installation by running following commands:

```
cd ~
git clone https://github.com/IntelRealSense/librealsense.git
cd librealsense
sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
```

- Apply udev changes within root:

```
sudo su
udevadm control --reload-rules && udevadm trigger
exit
```

- Add the following line to your .bashrc file to modify the path:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

- Apply the changes made so far by:

```
source ~/.bashrc
```

Dependencies

1. Install protobuf — Google's platform-neutral, language-neutral, extensible mechanism for serializing structured data by(in case of errors in this installatio go to next step 1.1. for binary installations):

```
cd ~
git clone --depth=1 -b v3.10.0 https://github.com/google/protobuf.git
cd protobuf
./autogen.sh
./configure
make -j1
sudo make install
cd python
export LD_LIBRARY_PATH=../src/.libs
python3 setup.py build --cpp_implementation
python3 setup.py test --cpp_implementation
sudo python3 setup.py install --cpp_implementation
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
sudo ldconfig
protoc --version
```

1.1. To install protobuf binaries:

```
sudo apt install protobuf-c*
sudo apt install python3-protobuf
```

1. Install C++ parallelism library libtbb-dev :

```
cd ~
wget https://github.com/PINTO0309/TBBonARMv7/raw/master/libtbb-dev_2018U2_armhf.deb
sudo dpkg -i ~/libtbb-dev_2018U2_armhf.deb
sudo ldconfig
rm libtbb-dev_2018U2_armhf.deb
```

RealSense SDK

- Install librealsense RealSense SDK:

```
cd ~/librealsense
mkdir  build  && cd build
cmake .. -DBUILD_EXAMPLES=true -DCMAKE_BUILD_TYPE=Release -DFORCE_LIBUVC=true
make -j1
sudo make install
```

- Let's just install Python bindings pyrealsense2 for librealsense`:

```
cd ~/librealsense/build
cmake .. -DBUILD_PYTHON_BINDINGS=bool:true -DPYTHON_EXECUTABLE=$(which python3)
make -j1
sudo make install
```

- Modify the PYTHONPATH environment variable by adding the following line to `.bashrc'

```
export PYTHONPATH=$PYTHONPATH:/usr/local/lib
```

- Apply the changes by:

```
source ~/.bashrc
```

- OpenGL installation:

```
sudo apt-get install python-opengl
sudo -H pip3 install pyopengl
sudo -H pip3 install pyopengl_accelerate==3.1.3rc1
```

- Change RPI settings to enable OpenGL :

```
sudo raspi-config
# "7. Advanced Options" – "A8 GL Driver" – "G2 GL (Fake KMS)"
```

ROS Driver

RealSense ROS driver in RPI is not availble from binaies, so let's install the driver from sources and its one depndency ddynamic_reconfiure . Follow the steps below for the installation.

- Create catkin_ws

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src/
git clone https://github.com/IntelRealSense/realsense-ros.git
git clone https://github.com/pal-robotics/ddynamic_reconfigure.git
cd realsense-ros/
git checkout `git tag | sort -V | grep -P "^2.\d+\.\d+" | tail -1`
cd ..
catkin_init_workspace
```

```
cd ..
catkin_make clean
catkin_make -DCATKIN_ENABLE_TESTING=False -DCMAKE_BUILD_TYPE=Release
catkin_make install
source ~/catkin_ws/devel/setup.bash
```

Usage

To use the installed driver run the following command:

```
roslaunch realsense2_camera rs_camera.launch enable_sync:=true
```

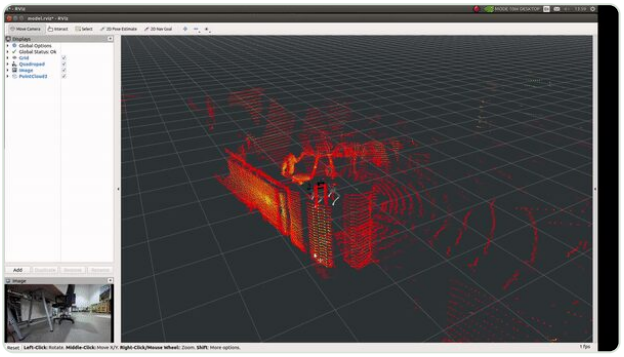
Now you can run rviz or some other tools to see the depth camera feed:

```
rosrun rviz rviz
```

Add a new frame of RGB image or depth one to see the result.

9 Simultaneous Localization & Mapping

Simultanous Localization and Mapping



_images/a1_point_cloud.gif

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.

mbs_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)

10 Navigation Configuration

Navigation Configuration

_images/a1_autonomous_navigation.gif

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

_images/emlid.jpg

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

> **Attention:** 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.

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 Simulation – Gazebo & Webots

Simulation

Unitree A1 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. Both simulators Gazebo and Webots are simulated with a fake node and visualization tool RViz is used.

Both simulators 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. Later Webots will be introduced which is quite robust, easy to use and is more realistic.

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

Webots Tutorials: https://cyberbotics.com/doc/guide/tutorials

Gazebo

qre_a1_gazebo & qre_controller

qre_a1_gazebo package contains the Gazebo simulation for the A1 robot. The robot can be teleoperated and can be used further for slam and autonomous navigation. This package is dependent upon qre_controller which contains several low level robot controllers state_estimator, quadruped_controller and contact_sensor .

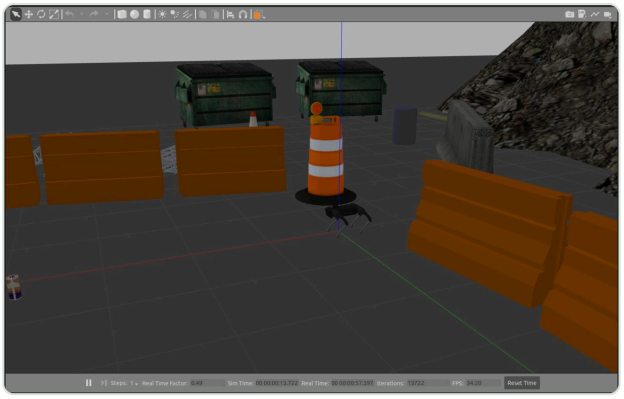
To launch Gazebo run the following command:

```
roslaunch qre_a1_gazebo gazebo.launch
```

This launch file runs all the necessary controllers and accept commands in cmd_vel(geometry_msgs/Twist) . The same package qre_controller can be used to verify the robot movements in RVIZ . To do so run the following command:

```
roslaunch qre_a1_description bringup.launch rviz:=true ll_control:=true
```

Now by running any teleop node the robot can be controlled.



QuadrupedRoboticsGazebo

Quadruped Robotics Gazebo simulation

unitree_gazebo & unitree_controller:

You can launch the Gazebo simulation by the following command:

```
roslaunch unitree_gazebo normal.launch rname:=a1 wname:=stairs
```

Where the rname means robot name, which can be laikago , aliengo or a1 . The wname means world name, which can be earth , space or stairs (for stairs world don't forget to change the respective world as described here). And the default value of rname is laikago , while the default value of wname is earth . In Gazebo, the robot should be lying on the ground with joints not activated.

Stand controller

After launching the gazebo simulation, you can start to control the robot:

```
rosrun unitree_controller unitree_servo
```

And you can add external disturbances, like a push or a kick:

```
rosrun unitree_controller unitree_external_force
```

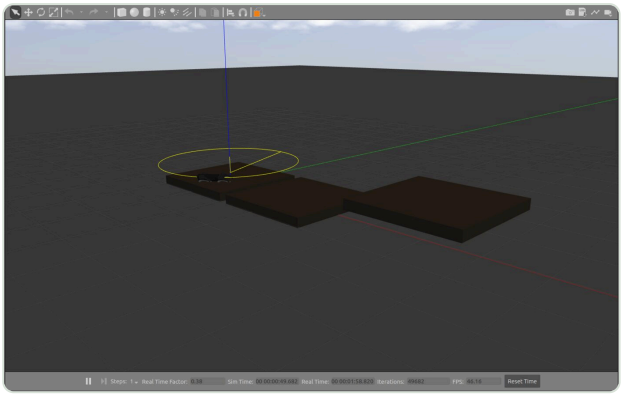
Position and pose publisher

Here we showed how to control the position and pose of robot without a controller, which should be useful in SLAM or visual development.

Then run the position and pose publisher in another terminal:

```
rosrun unitree_controller unitree_move_kinetic
```

The robot will turn around the origin, which is the movement under the world coordinate. And inside of the source file move_publisher , we also offered the method to move robot under robot coordinate. You can change the value of def_frame to coord::ROBOT and run the catkin_make again, then the unitree_move_publisher will move robot under its own coordinate.



Unitree Gazebo

Unitree Gazebo simulation

Webots

qre_a1_wbsim

qre_a1_wbsim contains Webots simulator file for Unitree A1. In the world folder different world and robot configurations can be found. Protos folder contains the proto(compact definition) of the robot.

Controller folder in a similar way contains two packages.

1. qre_a1_demo : This is a ros controller for Webots simulator. To run the Webots simulation with ROS follow the steps:

- Run Webots and Open the World file a1_demo_ros.wbt inside qre_a1_wbsim/worlds folder

```
webots --mode=pause --batch --stderr --stdout
```

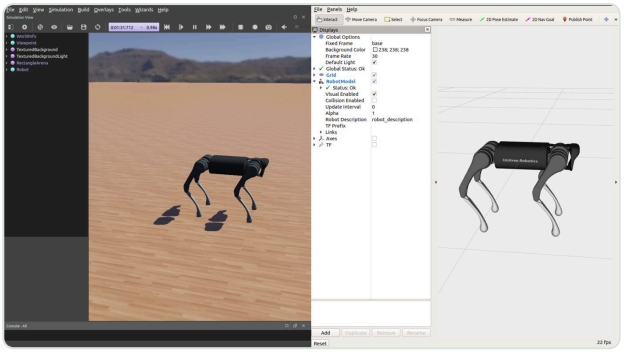
- Play Webots simulation (Ctrl 4) and run the ROS controller

```
roslaunch qre_wbs_controller wb_controller.launch
```

- For RVIZ visualization run the following command:

```
roslaunch qre_wbs_controller a1_rviz.launch
```

- The controller will launch a node: Node [webots_controller] Publications : wb_joint_states [sensor_msgs/JointState] Subscriptions : joint_states [sensor_msgs/JointState]



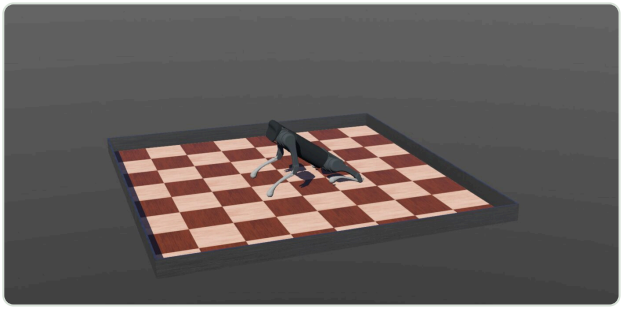
Webots ROS

Webots ROS communication

2. qre_wbs_demo : This package contains a bare minimum demo for how to control the robot using Webots api. The provided code should work out of the box and user do not need to have any knowledge of Webots api. Just sending right sequenced joint values and duration in pose_decomposition function should work. Similarly here again to run the world:

- Run Webots and Open the World file a1_demo.wbt inside qre_a1_wbsim/worlds folder

```
webots --mode=pause --batch --stderr --stdout
```



Webots demo

Webots planning demo

12 Augmentations – LiDAR & ZED2 Camera

Augmentations

Quadruped A1 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

_images/ouster.png

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

This procedure requires that the A1 has access to internet for initial installation if not done already.

Requirements

- Zed Camera
- Internet connection
- Access to the Nvidia onboard A1 (The connectors next to the A1'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

Updating URDF

The robot's upgradation also require the changes to be reflected in URDF. These changes can be made in qre_a1_description . The main to look at is xacro/accessories.xacro , here one can add joints, other xacro's etc. Also make sure to add the respective ENVIRONMENT VARIABLES in config/robot/qre_a1 from which by enable or disable the link the changes in the final URDF can be shown.

13 Changing Motor IDs

Changing Motor IDs

Requirements

- 1 x RS-485 Module : This module is crucial for enabling communication between your computer and the motor.
- 1 x Cable : A suitable RS-485 communication cable to connect the motor and RS-485 module.

Rs485 Module	Rs485 Module Cable

1. Overview of the A1 Motor

The A1 motor is a state-of-the-art, highly integrated permanent magnet synchronous motor (PMSM) designed specifically for robotics applications. Its compact design integrates several critical components that work together to deliver high performance and precise control.

A1 Motor

Key Components:

- Stator and Rotor : The stator houses the motor's windings, while the rotor is the moving part that generates motion.
- Drive Plate : This part controls the motor's operation by regulating power delivery.
- Reducer : A gearbox that reduces the motor's speed and increases torque output.
- Encoder : A sensor that provides feedback on the motor's position and speed, essential for precise control.
- Bearing : Ensures smooth rotation of the motor's moving parts.s

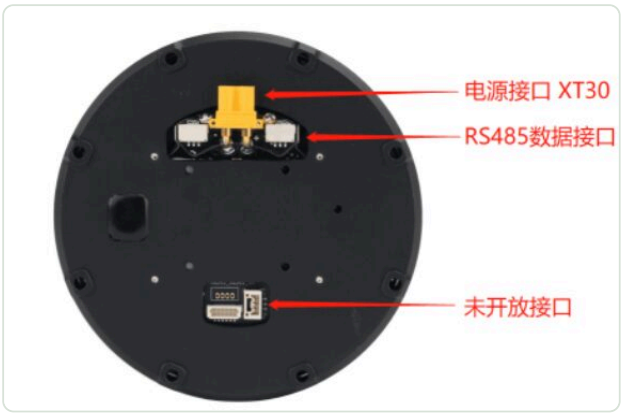
Key Parameters:

- Operating Voltage : 24V-30V

- Rated Voltage : 24V
- Maximum Torque : 33.5Nm
- Maximum Speed : 21 rad/s
- Communication Protocol : RS-485

This motor is specifically engineered for high-performance applications, such as in robotic systems, where reliability, precision, and power efficiency are critical.

2. RS-485 Interface Connection

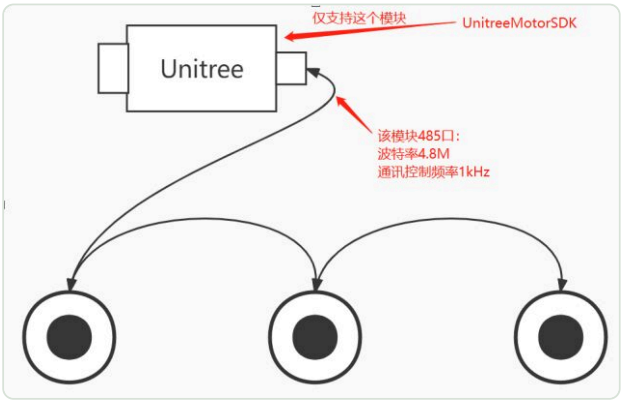


A1 Motor

The RS-485 interface is a standard communication protocol used in industrial and robotic applications due to its robustness in noisy environments and ability to support long-distance communication.

Connecting the A1 Motor:

- The A1 motor features two equivalent RS-485 interfaces, allowing for the daisy-chaining of multiple motors. This means you can control up to three motors in series using a single RS-485 line.



A1 Motor

- To control the motor from a computer, connect the RS-485 interface of the motor to your computer via a USB-to-RS-485 converter.

- Power Supply : Ensure that a 24V DC power supply is connected to the motor. Once powered, the motor's green indicator light will blink, signaling that it is ready for operation.

> **Note:** The communication frequency for controlling the A1 motor is up to 3 kHz, which allows for fast and responsive motor control, essential in dynamic robotic applications.

3. Configuring Motor IDs

In a multi-motor setup, each motor must be uniquely identified on the RS-485 communication line. This is achieved by assigning each motor a unique ID.

Each motor connected via the RS-485 serial port must have a unique ID for individual control. The following instructions explain how to modify the motor ID using the SDK.

I. Clone and Build the SDK

First, clone the Unitree Actuator SDK (A1B1 branch) and build it.

Steps to Clone and Build:

1. Open a terminal window.
2. Clone the repository: git clone -b A1B1 https://github.com/unitreerobotics/unitree_actuator_sdk.git
3. Navigate to the SDK directory: cd unitree_actuator_sdk
4. Create a build directory and compile the code: mkdir build cd build cmake .. make

After successful compilation, you should see an executable file named changeID in the build directory.

II. USB to RS-485 Connection

The A1 robot motors are controlled via an RS-485 serial port, which is accessed through a USB-to-RS-485 module. This module allows communication between the motors and your computer.

Steps to Check the Serial Port Name:

1. Connect the USB-to-RS-485 module to your computer.
2. On Linux, the system will assign a serial port name starting with ttyUSB . To identify this port, run the following commands: cd /dev ls | grep ttyUSB

The command will display the connected serial port, e.g., /dev/ttyUSB0 .

III. Changing Motor ID

Each motor needs a unique ID for control. You will modify the motor ID using the changeID program from the SDK.

Steps to Change Motor ID:

1. Navigate to the build directory where the changeID executable is located: cd build
2. Run the changeID program with administrator privileges: sudo ./changeID You will be prompted to enter the administrator password. After entering the password, the program will start.
3. Enter the serial port name (e.g., /dev/ttyUSB0) when prompted. This will put all motors into “ID modification mode”. In this mode, the motor output shaft behaves like an electronic ratchet, meaning you will feel resistance when rotating it manually.
4. Use the following steps to set the motor ID: Rotate the motor output shaft using the provided tooling (M4 screws with a depth of no more than 5mm). The motor ID will be set based on the number of turns: One turn sets the ID to 0 (Hip Motor). Two turns set the ID to 1 (Thigh Motor). Three turns set the ID to 2 (Calf Motor). After setting the ID, type a in the terminal and press Enter to save the motor ID.

Note: The “turns” referred to here are not full rotations but the distinct incremental steps (jerks) you will feel while rotating the motor shaft in ID modification mode. These jerks indicate the motor is in the electronic ratchet mode, and each jerk corresponds to a new ID being assigned.

Important: Ensure the tooling screws are M4 and inserted no deeper than 5mm into the motor shaft. The IDs can only be 0 , 1 , or 2 .

Following these steps will allow you to successfully modify the motor IDs for Unitree A1 Robot motors. For any further issues or troubleshooting, refer to the Unitree Actuator SDK documentation.

4. Testing Motor Configuration

After assigning and saving the motor IDs, it’s essential to verify that the motors are correctly configured and operational.

Verifying Motor Operation:

1. Modify and Compile Test Program: - The SDK includes a test program named main.cpp . Before running this program, open the code and set the correct motor ID and serial port name. - Compile the program by navigating to the SDK directory and using the following commands:

```
mkdir build
cd build
cmake ..
make
sudo ./main
```

1. Running the Test: - Execute the compiled test program. This program will command the motor to perform basic operations such as rotating to specific positions. - The program will display

important motor parameters on your screen, including position, temperature, and any error codes that might indicate issues with the motor or its configuration.

5. Troubleshooting

In the event of issues during the configuration process, the following are common problems and their solutions:

Common Issues and Solutions:

- Incorrect Serial Port or Missing Adapter : Ensure that the correct serial port is selected, and the USB-to-RS-485 adapter is properly connected.
- Insufficient Privileges : If you encounter permission issues, rerun the program with sudo to ensure it has the necessary access.
- Incorrect Motor ID : Double-check that each motor has been assigned a unique ID and that the ID matches what you specified in the test program.
- No Power Supply : Confirm that the 24V power supply is connected and the green indicator light on the motor is blinking.
- Communication Failure : Check all connections between the motor, RS-485 module, and computer to ensure there are no loose or faulty wires.

Additional Resources : For more detailed troubleshooting, consult the SDK documentation or the motor's manual. These resources provide in-depth information on advanced configurations and debugging.

This guide provides comprehensive steps for configuring motor IDs on the A1 robot. By following these instructions carefully, you can ensure your motors are correctly identified and ready for precise control in your robotic application. For any advanced configurations or troubleshooting beyond this guide, refer to the documentation provided with the A1 motor or seek support from the manufacturer.

14 Robot Models – A1, Laikago Pro & Aliengo

Robot Models

Unitree Robotics have launched several models depending on different configuration. Mainly three models some in different configurations are available.

A1

A1 has an optimal design due to which it can not reduce the cost but make the service life and motion performance better. The robot can be controlled with Android and IOS apps as well as ROS.

It has 2 modes to work:

1. Normal mode
2. Sports mode

In sports mode the robot can get the maximum performance and can run upto 3.3 m/s speed, having 33.5 NM torque. With that much power the robot can do back flips quite easily.

Moreover the robot is equipped with a depth camera with transmission quality of 720P with 30 frames per second. The robot can be equipped with a lidar and NVIDIA TX2 boards for operations like slam, autonomus navigation and obstacle avoidance.

Laikago Pro

Laikago is the first version and the most powerful robot in Unitree Robotics product family. It has 12 strong srong motors which features 18KW of the total maximum instantenous power as well as 0.8KW/KG of power density almost double of supercar(0.45KW/KG). It has excellent stability and adapt to external disturbances in certain amplitude.

Laikago

Note: Laikago production has come to an end and will not be produced anymore.

Aliengo

Aliengo has a quite slim and versatile design. It has depth perception vision system, high explosive sports performance, adaptive complex terrain, long battery life with heavy load and built-in intelligent air cooling system.

The robot can work in 2 modes:

1. Normal mode
2. Sports mode.

Aliengo can walk, run and even go up and down stairs with a stir height between 16-18cm. The robot features a quite powerful vision system with which 3d construction, dynamic obstacle perception, huamn posture recognition and tracking with face recognition.

Aliengo

15 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 .

16 Getting Involved

Getting Involved

The software provided are Opensource, so contributions from community are welcome on our Github repository

17 FAQ

Frequently Asked Questions

Wireless Access A1

The A1 can be wirelessly accessed by its onboard hotspot. It usually has the name UnitreeRoboticsA1-## 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 A1

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 moetherboard goes to sleep, so if you connect an external mouse or keyboard you can directly wake it up.

No Video Stream A1

The Real-sense device on board A1 is connected to the Nvidia board so Intel real-sense must run from the Nvidia board.

There is no ROS driver is running by default on the board.To get it to work:

1. Run the perception driver on Nvidia board.
2. To get all the camera feed to your remote PC either use:

- ROS multi machine communication (ROS network setup) and get everything from Nvidia to your remote PC.
- You can export the ports from your Nvidia board and use on your remote PC.The Turtlebot preception guide can be followed whith slight modification in device configuration.