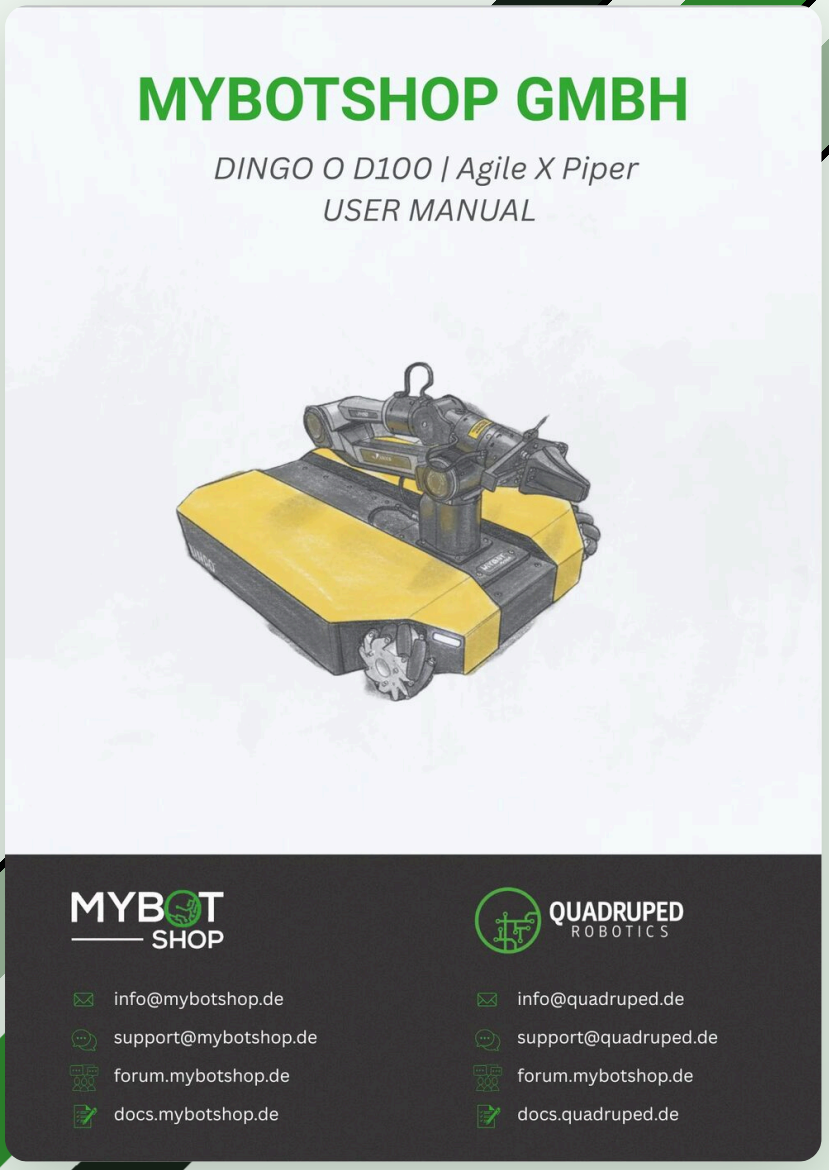
MYBOTSHOP

MOBILE MANIPULATION

Dingo D100

Mobile Manipulator – User Manual



CONTACT

info@mybotshop.de

support@mybotshop.de

mybotshop.de

Contents

1 Attention

Attention: Read this document carefully before operating the Dingo D100. In case of doubt, consult MYBOTSHOP directly before putting the robot — and especially the AgileX Piper arm — into operation. Contact information is available in the final chapter, Getting Help & RMA.

Do not allow persons who are not familiar with mobile manipulators or these instructions to operate the robot. Robotic arms 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 emergency-stop procedure (Chapter 4, Quick Start) before powering the platform on.

Caution: Non-technical persons are not permitted to operate the Piper arm — the damage it can cause is severe. Always keep the workspace clear, use slow speeds while testing, and keep the emergency stop within reach.

2 Disclaimer

The Dingo D100 documented here is a MYBOTSHOP mobile manipulation platform combining a Clearpath Dingo-O omnidirectional base with an AgileX Piper 6-DOF robotic arm. Basic ROS 2 understanding is required to operate the software stack covered in this manual — if you are not familiar with ROS 2, we recommend checking the ROS 2 Jazzy documentation first.

The client acknowledges and agrees that any information or materials provided by MYBOTSHOP 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 research and development use only. Users should avoid making dangerous modifications or using the robot in hazardous ways. This manual documents the Dingo Agile (D100) configuration — Dingo-O base, AgileX Piper arm, ROS 2 Jazzy — and separately flags the related Dingo Lite configuration, which is a different hardware and software stack; always confirm which configuration your unit is before following a procedure.

MYBOTSHOP 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 Dingo D100

Dingo D100 Mobile Manipulator

MYBOTSHOP Dingo Mobile Manipulation Platforms

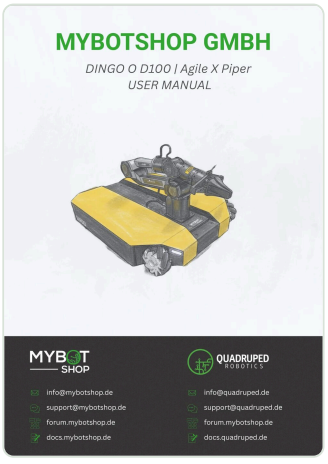
The MYBOTSHOP Dingo series consists of mobile manipulation platforms built on the Clearpath Dingo base with integrated robotic arms. These platforms are designed for indoor research and development applications requiring both mobility and manipulation capabilities.

Available Configurations:

- Dingo Lite (L) : Dingo-D differential base + Ufactory Lite6 arm (ROS Noetic)
- Dingo Agile (A) : Dingo-O omnidirectional base + AgileX Piper arm (ROS2 Jazzy)

Current Software Version: v3.6 (December 2025)

Introduction



MYBOTSHOP Dingo D100 with AgileX Piper Arm

The MYBOTSHOP Dingo D100 is a mobile manipulation platform combining the Clearpath Dingo-O omnidirectional base with the AgileX Piper 6-DOF robotic arm. This platform is designed for research and development applications requiring both mobility and manipulation capabilities.

Quick Links

Resource	Link
MYBOTSHOP Website	mybotshop.de
Dingo D100 Documentation	docs.mybotshop.de
Robot Webserver	192.168.131.1:9000
ROS2 Jazzy Documentation	docs.ros.org/jazzy
Clearpath Dingo Docs	docs.clearpathrobotics.com
Nav2 Documentation	nav2.org

External Dependencies

Package	Documentation
Clearpath Dingo	dingo
Intel RealSense ROS2	realsense-ros
Teleop Twist Keyboard	teleop_twist_keyboard
Nav2 Navigation	navigation2
MoveIt2	moveit2

Current Available Features

Feature	Status	Notes
Dingo Visualization (RViz2)	Available	
Dingo Base Driver	Available	Joint State, IMU, CMD, Odometry
Dingo Description (URDF/Xacro)	Available	
Hokuyo LiDAR	Available	
Intel RealSense D405	Available	Wrist-mounted on Piper arm
Piper Arm Driver	Available	
MoveIt2 Motion Planning	Available	
Autostart Services	Available	
Webserver	Available	
SLAM Navigation	Available	
Map Navigation	Available	

Manuals

Note: Product documentation and manuals are available for customers only. Please contact MYBOTSHOP for access to additional documents.

Available Resources:

- Dingo Technical Specification
- Dingo User Manual
- AgileX Piper Documentation

4 Quick Start – Network, Webserver & Power

Pre-requisites

Before operating the Dingo D100, ensure you have:

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

Robot App

The Dingo D100 comes with pre-installed ROS2 software. The robot's onboard computer runs Ubuntu 24.04 LTS with ROS2 Jazzy. Core drivers and services start automatically on boot via systemd services.

Robot Webserver

The Dingo D100 features a web-based control interface accessible at:

URL: http://192.168.131.1:9000

Default Credentials:

- Username: admin
- Password: mybotshop

The webserver provides:

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

Charging Robot

To charge the Dingo D100:

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

Emergency Stop

The Dingo D100 is equipped with emergency stop buttons. When pressed:

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

To resume operation after emergency stop:

1. Clear the emergency condition
2. Release the emergency stop button
3. Restart the platform services if needed

Power On

To power on the Dingo D100:

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

Power Off

To safely power off the Dingo D100:

1. Stop all running applications
2. Press the main power switch to turn off

Basic Operation

Remote Control

The Dingo D100 supports multiple teleoperation methods:

1. Logitech Controller - Highest priority, requires deadman switch
2. Web Interface - Browser-based joystick control
3. Keyboard Teleop - ROS2 teleop_twist_keyboard

Network Interface

Network Table

IP Address	Device	Username	Password

192.168.131.1	Dingo Computer	robot	clearpath
192.168.131.1:9000	Webserver	admin	mybotshop
192.168.131.2	Dingo MCU		
192.168.131.20	Hokuyo LiDAR		

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

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

Ethernet Connection

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

Static Network Connection

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

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

Verify the connection:

```
# Check your local IP
ifconfig

# Ping the robot
ping 192.168.131.1

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

5 Safety Guidelines

Robotic Manipulator Safety

When deploying robotic manipulators, it is imperative to prioritize safety procedures to mitigate risks and ensure secure operations. The following guidelines delineate key safety measures when working with robotic manipulators.

Work Area Safety

- Maintain a clean and well-organized work area. Cluttered or inadequately lit spaces can impede the proper functioning of sensors and hinder precise manipulation.
- Avoid operating robotic manipulators in hazardous environments, such as those containing corrosive substances, extreme temperatures, or sharp objects that may damage the manipulator.
- Ensure that only authorized personnel are present in the vicinity during manipulator operation to prevent interference and ensure a safe working environment.

Electrical Safety

- Ensure the manipulator's power system adheres to electrical safety standards. Regularly inspect and maintain power components to prevent malfunctions.
- Implement safeguards to protect the manipulator from adverse environmental conditions, such as exposure to moisture or extreme temperatures.
- Regularly inspect power cables and connections, promptly replacing damaged components to minimize the risk of electrical issues.

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 with personnel or other objects.
- Regularly calibrate and test the manipulator's sensors and systems to ensure precise and reliable performance during manipulation tasks.

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 for handling unexpected situations during manipulator operation.

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.

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.

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 demand correct usage to avoid accidents and ensure a secure environment. Please adhere to proper procedures diligently, prioritizing both precision and safety.

Autonomous Robot Safety

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:

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.

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.

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.

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.

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.

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.

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.

Robot Maintenance

Regular maintenance ensures optimal robot performance and longevity. Follow the manufacturer's guidelines for maintenance schedules and procedures.

6 ROS 2 Software – Autostart, Webserver & Teleoperation

Autostart

Autostart Configuration

Configure the Dingo ROS 2 nodes to start automatically on boot.

MYBOTSHOP Platform Service

If using the MYBOTSHOP configuration, services are managed via systemd:

Start Platform Service:

```
sudo service dingo-platform start
```

Check Service Status:

```
sudo service dingo-webserver status
sudo service dingo-platform status
```

Service Status Indicators:

- Green : Service is running correctly
- Red : Service has failed - restart needed
- Grey : Service has not started yet

Restart Service:

```
sudo service dingo-platform restart
```

Update Startup Job:

After modifying launch files, update the startup job:

```
ros2 run dingo_bringup startup_installer.py
```

Note: If the service is green, do not launch dingo_bringup manually as it's already running in the background.

Manual Launch

Launch the Dingo ROS driver manually:

```
ros2 launch dingo_bringup system.launch.py
```

Custom systemd Service

Create a custom service file:

```
sudo nano /etc/systemd/system/dingo-ros2.service
```

Add the following content:

```
[Unit]
Description=Dingo ROS 2 Base Driver
After=network.target

[Service]
Type=simple
User=robot
Environment="ROS_DOMAIN_ID=0"
ExecStart=/bin/bash -c "source /opt/ros/jazzy/setup.bash && source
/opt/mybotshop/install/setup.bash && ros2 launch dingo_bringup system.launch.py"
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Enable Service:

```
sudo systemctl daemon-reload
sudo systemctl enable dingo-ros2.service
sudo systemctl start dingo-ros2.service
```

Check Status:

```
sudo systemctl status dingo-ros2.service
```

View Logs:

```
journalctl -u dingo-ros2.service -f
```

Webserver

Robot Webserver

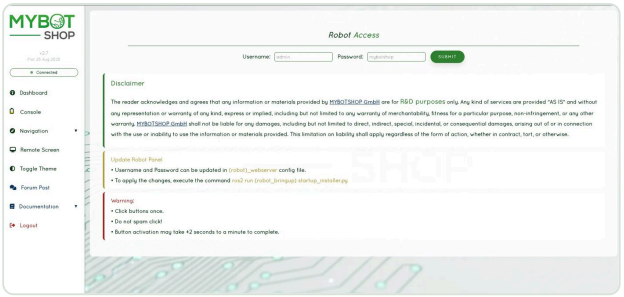
The Dingo D100 features a web-based control interface built with Flask.

Access: http://192.168.131.1:9000

Default Credentials:

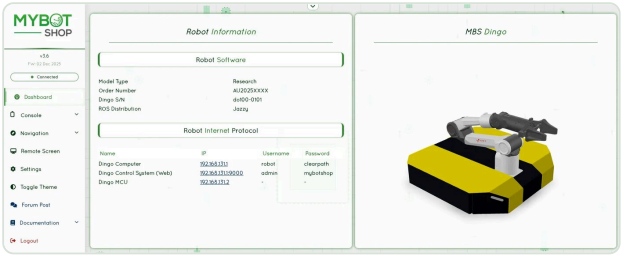
- Username: admin
- Password: mybotshop

Login



Webserver Login Page

Dashboard

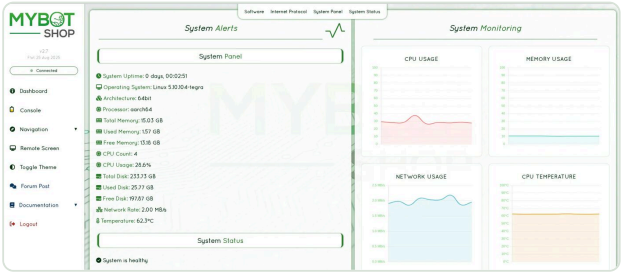


Main Dashboard with 3D Robot Visualization

Features:

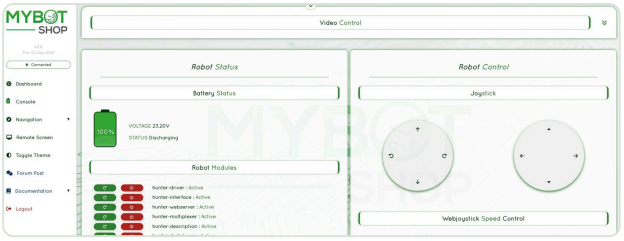
Feature	Description
3D Visualization	Real-time robot model via GLTF
Joystick Teleop	Browser-based joystick control
Service Management	Start/stop ROS2 nodes and services
Map Visualization	View navigation maps
Battery Monitoring	Real-time battery state
VNC Access	Remote desktop via VNC

System Management



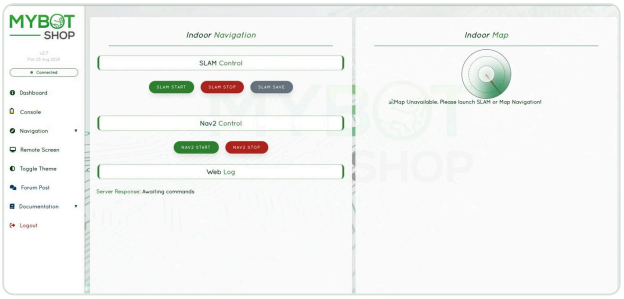
System Management Interface

Console



Web Console Interface with Joystick Control

Navigation



Indoor Navigation View with SLAM and Nav2 Controls

VNC Remote Desktop

VNC Remote Desktop Access

Configuration (config/robot_webserver.yaml):

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

Launch Webserver:

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

```
ros2 launch dingo_webserver webserver.launch.py
```

VNC Setup:

To enable VNC remote desktop:

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

Teleoperation

Keyboard Teleoperation

Control the Dingo using keyboard:

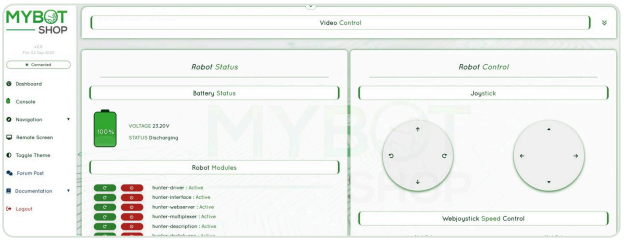
```
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  --ros-args --remap cmd_vel:=/do100_0101/cmd_vel
```

Keyboard Controls:

- i : Forward
- k : Stop
- j : Turn left
- l : Turn right
- , : Backward
- u : Forward-left curve
- o : Forward-right curve

Web Joystick

If using the MYBOTSHOP webserver, a web-based joystick is available:



Web console with joystick control

Joystick Teleoperation

Install joystick packages:

```
sudo apt install ros-jazzy-joy ros-jazzy-teleop-twist-joy
```

Launch joystick teleoperation:

```
ros2 launch teleop_twist_joy teleop-launch.py
```

Logitech Controller

The Dingo can be controlled via the Logitech controller:

- Ensure the Logitech dongle is connected to the Dingo PC (192.168.131.1)
- Do not use the dongle with other computers

Custom Velocity Commands

Send velocity commands directly:

```
ros2 topic pub /do100_0101/cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.5}, angular: {z: 0.0}}"
```

Note: Recommended teleoperation speed: 0.2 m/s for mapping.

7 Visualization & Hardware Rigs

Visualization

RViz2 Visualization

Launch RViz2 with Dingo model:

```
ros2 launch dingo_viz view_robot.launch.py
```

This displays:

- Robot URDF model
- TF frames
- Coordinate axes
- Sensor data (if available)

View Robot Model Only

To view just the robot model without running the base driver:

```
ros2 launch dingo_description display.launch.py use_rviz:=true
```

Web-Based Visualization

For remote visualization, use the web interface:

Remote desktop access for visualization

Custom RViz Configuration

Save your RViz configuration:

1. Configure displays as desired
2. File > Save Config As > my_dingo_config.rviz

Load custom configuration:

```
rviz2 -d my_dingo_config.rviz
```

TF Tree

View the transform tree:

```
ros2 run tf2_tools view_frames
```

This generates a PDF showing the complete TF tree structure.

Foxglove Studio

For modern web-based 3D visualization:

1. Install Foxglove Bridge: sudo apt install ros-jazzy-foxglove-bridge
2. Launch the bridge: ros2 launch foxglove_bridge foxglove_bridge_launch.xml
3. Open Foxglove Studio and connect to your robot

Rigs

Hardware Configurations

The Dingo D100 can be configured with different sensor and accessory rigs.

Standard Configuration

- Clearpath Dingo-O omnidirectional base
- AgileX Piper 6-DOF arm
- Hokuyo LiDAR (192.168.131.20)
- Intel RealSense D405 (wrist-mounted)

Adding Custom Sensors

The Dingo provides multiple interfaces for custom sensors:

- USB ports : For cameras and peripherals
- Ethernet : For IP-based sensors
- 24V Power : For powered accessories

Mounting Options

Use the Dingo's mounting rails to attach additional hardware:

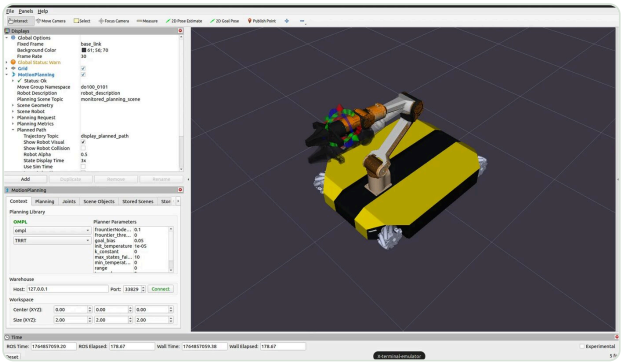
- T-slot aluminum rails
- 80/20 compatible mounting brackets
- Custom 3D printed mounts

8 Manipulation – AgileX Piper Arm

Manipulation

AgileX Piper Arm

The Dingo D100 is equipped with an AgileX Piper 6-DOF robotic arm.



RViz2 with MoveIt2 Motion Planning for Piper Arm

Launch Piper Driver

Start the Piper arm driver:

```
ros2 launch dingo_manipulation piper.launch.py
```

MoveIt2 Planning

Launch MoveIt2 for motion planning:

```
ros2 launch dingo_manipulation piper_moveit.launch.py
```

MoveIt2 Features:

- Motion planning with OMPL
- Collision detection
- Path visualization
- Scene management

Piper Gripper

Control the Piper gripper:

Open Gripper:

```
ros2 topic pub /piper/gripper_cmd std_msgs/msg/Float64 "data: 0.0"
```

Close Gripper:

```
ros2 topic pub /piper/gripper_cmd std_msgs/msg/Float64 "data: 1.0"
```

Joint Control

Move individual joints:

```
ros2 topic pub /piper/joint_trajectory_controller/joint_trajectory \
  trajectory_msgs/msg/JointTrajectory \
  "{joint_names: ['joint1', 'joint2', 'joint3', 'joint4', 'joint5', 'joint6'], \
  points: [{positions: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0], time_from_start: {sec: 2}}]}"
```

Safety Guidelines

> **Warning:** Non-technical persons are not permitted to use the robotic arm as the damage it can cause is severe. Always follow safety guidelines when operating the manipulator.

- Always ensure the workspace is clear before operating
- Use slow speeds during testing
- Keep emergency stop accessible
- Follow the safety guidelines in the Safety section

9 Navigation

Navigation

Simultaneous Localization and Mapping

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

Prerequisites:

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

Launch SLAM:

```
ros2 launch dingo_navigation slam.launch.py
```

Mapping Procedure:

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

Save the Map:

Once satisfied with the map, save it:

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

Rebuild Workspace (if using custom map name):

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

Odometry Navigation

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

Launch Odometry Navigation:

```
ros2 launch dingo_navigation odom_navi.launch.py
```

This mode is useful for:

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

Map Navigation

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

Prerequisites:

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

Launch Map Navigation:

```
ros2 launch dingo_navigation map_navi.launch.py
```

Setting Goals:

Goals can be set via:

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

Configuration Files:

- param/nav2_slam.yaml - SLAM parameters
- param/nav2_odom.yaml - Odometry navigation parameters
- param/nav2_map.yaml - Map navigation parameters

Dependencies:

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

For more information, see the Nav2 documentation .

10 Sensors & Simulation

Sensors

Built-in Sensors

The Dingo D100 includes:

- Wheel Encoders : For odometry calculation
- IMU : Internal measurement unit

Supported External Sensors

The Dingo's expansion interfaces support various sensors:

LiDAR:

- Velodyne VLP-16
- Ouster OS1
- SICK LMS series
- RPLidar
- Hokuyo

Cameras:

- Intel RealSense D405/D435/D455
- ZED/ZED2 Stereo Camera
- USB webcams

IMU:

- Drotek IMU
- Xsens MTi series
- Microstrain
- Phidgets IMU

GPS:

- u-blox receivers
- Emlid Reach
- Fixposition

Hokuyo LiDAR

The Hokuyo LiDAR provides 2D laser scan data.

Launch Hokuyo:

```
ros2 launch dingo_lidars hokuyo.launch.py
```

Configuration:

- IP Address: 192.168.131.20
- Provides 2D laser scan data

Intel RealSense D405

The Intel RealSense D405 depth camera is mounted on the Piper arm wrist.

Launch RealSense:

```
ros2 launch dingo_depth_camera realsense_d405.launch.py
```

Note: The RealSense camera provides RGB and depth data for manipulation tasks.

Sensor Topics

Common sensor topics:

- /scan : 2D LiDAR scan
- /camera/color/image_raw : RealSense color image
- /camera/depth/image_rect_raw : RealSense depth image
- /imu/data : IMU data

Simulation

Gazebo Simulation

The Dingo D100 can be simulated in Gazebo Harmonic.

Prerequisites:

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

Launch Simulation:

```
ros2 launch dingo_gz_sim simulation.launch.py
```

This launches:

- Gazebo Harmonic with the Dingo model
- Robot state publisher
- ros_gz bridge for topic communication

Simulation Topics

The simulation bridges the following topics:

- /cmd_vel : Velocity commands
- /odom : Odometry data
- /scan : Simulated LiDAR
- /tf : Transform data
- /joint_states : Joint state information

Custom Worlds

Load a custom world:

```
ros2 launch dingo_gz_sim simulation.launch.py world:=custom_world.sdf
```

Adding Objects

Spawn objects in the simulation using Gazebo's GUI or via command line:

```
gz service -s /world/default/create \
  --reqtype gz.msgs.EntityFactory \
  --reptype gz.msgs.Boolean \
  --req 'sdf: "<sdf>...</sdf>"'
```

11 Hardware Specification

Specifications

Mechanical

Parameter	Value
Base Platform	Clearpath Dingo-O
Drive System	Omnidirectional (4 mecanum wheels)
Manipulator	AgileX Piper 6-DOF
Gripper	Piper parallel gripper

Electrical

Parameter	Value
Battery Type	Lithium-ion
Nominal Voltage	24V
Communication Interface	Ethernet, USB

Base Software

Parameter	Value
Operating System	Ubuntu 24.04 LTS
ROS Distribution	ROS2 Jazzy
Navigation Stack	Nav2
Motion Planning	MoveIt2

12 ROS 2 Package Reference, Debugging & Installation

Packages

Core Packages

The Dingo D100 ROS2 software consists of the following packages:

dingo_bringup

Main launch package for the Dingo platform.

Launch Files:

- system.launch.py : Full system bringup
- base.launch.py : Base driver only

Usage:

```
ros2 launch dingo_bringup system.launch.py
```

dingo_description

URDF/Xacro robot description package.

Launch Files:

- display.launch.py : Display robot model in RViz2

Usage:

```
ros2 launch dingo_description display.launch.py use_rviz:=true
```

dingo_viz

Visualization package with RViz2 configurations.

Launch Files:

- view_robot.launch.py : View robot with sensors

Usage:

```
ros2 launch dingo_viz view_robot.launch.py
```

dingo_navigation

Navigation package with Nav2 configurations.

Launch Files:

- slam.launch.py : SLAM mapping
- odom_navi.launch.py : Odometry-based navigation
- map_navi.launch.py : Map-based navigation

Configuration:

- param/nav2_slam.yaml
- param/nav2_odom.yaml
- param/nav2_map.yaml

dingo_webserver

Web-based control interface.

Launch Files:

- webserver.launch.py : Start the Flask webserver

Configuration:

- config/robot_webserver.yaml

dingo_manipulation

Piper arm manipulation package.

Launch Files:

- piper.launch.py : Piper arm driver
- piper_moveit.launch.py : MoveIt2 planning

Package Installation

Install from source:

```
cd /opt/mybotshop/src
git clone <repository_url>
cd /opt/mybotshop
colcon build --symlink-install
source install/setup.bash
```

Debugging

Common Issues

Robot not responding to commands:

1. Check service status: sudo service dingo-platform status
2. Verify network connection: ping 192.168.131.1
3. Check ROS2 topics: ros2 topic list

No sensor data:

1. Verify sensor connections
2. Check sensor-specific topics
3. Review sensor launch files

Diagnostic Commands

Check ROS2 Topics:

```
ros2 topic list
ros2 topic echo /do100_0101/odom
```

Check ROS2 Nodes:

```
ros2 node list
ros2 node info /dingo_base
```

Check TF Tree:

```
ros2 run tf2_tools view_frames
```

Check Service Status:

```
sudo systemctl status dingo-platform
journalctl -u dingo-platform -f
```

Network Debugging

Verify IP Configuration:

```
ifconfig
ip addr show
```

Check ROS2 Domain:

```
echo $ROS_DOMAIN_ID
```

Test Connectivity:

```
ping 192.168.131.1
ping 192.168.131.2
```

Log Files

View System Logs:

```
journalctl -u dingo-platform -f
journalctl -u dingo-webserver -f
```

ROS2 Logging:

```
ros2 run rqt_console rqt_console
```

Restarting Services

Restart All Services:

```
sudo service dingo-platform restart
sudo service dingo-webserver restart
```

Rebuild Workspace:

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

Miscellanious

SSH Access

Connect to the robot via SSH:

```
ssh -X robot@192.168.131.1
# Password: clearpath
```

File Transfer

Transfer files to/from the robot:

```
# Copy to robot
scp local_file.txt robot@192.168.131.1:~/

# Copy from robot
scp robot@192.168.131.1:~/remote_file.txt ./
```

Remote Desktop

Access the robot desktop via VNC through the webserver:

1. Navigate to http://192.168.131.1:9000
2. Click “Remote Screen” in the sidebar
3. Connect using VNC password: mybotshop

ROS2 Environment Variables

Common environment variables:

```
export ROS_DOMAIN_ID=0
export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
export DINGO_NS=do100_0101
```

Updating Software

Update the ROS2 workspace:

```
cd /opt/mybotshop/src
git pull
cd /opt/mybotshop
colcon build --symlink-install
source install/setup.bash
```

Backup Configuration

Backup important configuration files:

```
tar -czvf dingo_backup.tar.gz \
  /etc/clearpath/robot.yaml \
  /opt/mybotshop/src/mybotshop
```

Factory Reset

To restore factory settings, contact MYBOTSHOP support for the recovery image.

Installation

ROS2 Jazzy Installation

The Dingo D100 comes pre-installed with ROS2 Jazzy. For external PC setup:

Prerequisites:

- Ubuntu 24.04 LTS
- ROS2 Jazzy

Install ROS2 Jazzy:

```
sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

sudo apt install software-properties-common
sudo add-apt-repository universe
sudo apt update && sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo
```

```
$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
sudo apt update && sudo apt upgrade -y
sudo apt install ros-jazzy-desktop
```

Source ROS2:

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

Clearpath Setup

For Clearpath Dingo setup, use the computer installer:

```
wget -c
https://raw.githubusercontent.com/clearpathrobotics/clearpath_computer_installer/main/clear
path_computer_installer.sh
bash -e clearpath_computer_installer.sh
```

Robot Configuration

Edit the robot configuration file:

```
sudo nano /etc/clearpath/robot.yaml
```

Example configuration:

```
serial_number: do100-0101
version: 0
system:
  hosts:
    - hostname: cpr-do100-0101
      ip: 192.168.131.1
  ros2:
    namespace: do100_0101
platform:
  controller: logitech
```

Workspace Setup

Create a ROS2 workspace:

```
mkdir -p ~/dingo_ws/src
cd ~/dingo_ws
colcon build --symlink-install
source install/setup.bash
```

Add to bashrc:

```
echo "source ~/dingo_ws/install/setup.bash" >> ~/.bashrc
```

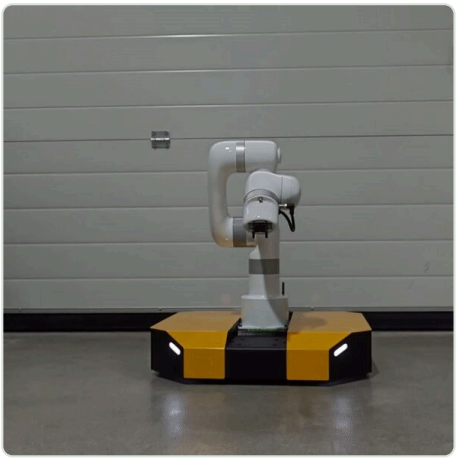
Dependencies

Install common dependencies:

```
sudo apt install ros-jazzy-navigation2 ros-jazzy-nav2-bringup
sudo apt install ros-jazzy-slam-toolbox
sudo apt install ros-jazzy-teleop-twist-keyboard ros-jazzy-teleop-twist-joy
sudo apt install ros-jazzy-foxglove-bridge
```

13 Hardware Variant – Dingo Lite (Dingo-D + Ufactory Lite6, ROS Noetic)

Dingo Lite (Dingo L)



Dingo Lite with Ufactory Lite6 Arm

The Dingo Lite is a mobile manipulation platform combining the Clearpath Dingo-D differential drive base with the Ufactory Lite6 robotic arm. This platform runs ROS Noetic on Ubuntu 20.04.

Key Features:

- Differential drive mobility
- Ufactory Lite6 6-DOF arm with gripper
- ROS Noetic support
- Hokuyo LiDAR
- Intel RealSense D405 (wrist-mounted)

Autostart

Autostart Configuration

The Dingo-D ROS drivers are automatically launched during the robot's startup process.

Startup Job Installation

To install or update the startup job:

- Give root permission to the installation script:

```
sudo chmod +x startup_installer.py
```

- Run the script from the terminal:

```
./startup_installer.py
```

- Provide the workspace name as ros_ws and the directory as /home/administrator

> **Note:** If the Dingo-D's PC username is not administrator, manual changes will have to be made in the startup job.

ROS Environment Setup

Add a setup.bash file in /etc/ros if it's not there already:

```
cd
sudo touch /etc/ros/setup.bash
```

Edit the file and add the configuration:

```
source /home/administrator/ros_ws/devel/setup.bash

export ROBOT_SETUP=/etc/ros/setup.bash
export DINGO_OMNI=0
export DINGO_JOY_DEV=/dev/input/js0
```

Base Driver Installation

Source and install the Dingo-D base driver:

```
source /etc/ros/setup.bash
roscd mmp_bringup/scripts
./can-udp-bridge.sh
./install
```

Webserver

Web Interface

The Noetic version does not include a web-based control interface.

For remote operation, use SSH and ROS command-line tools:

```
ssh -X administrator@192.168.131.1
```

Launch visualization remotely:

```
roslaunch mmp_viz view_robot.launch
```

Teleoperation

Keyboard Teleoperation

To operate the Dingo-D either in simulation or the real robot:

```
rosrun key_teleop key_teleop.py
```

Joystick Teleoperation

The Dingo-D can be controlled via a joystick connected to the robot.

Important: The Logitech controller dongle should be inserted and used with the Dingo PC (192.168.131.1). Do not use the dongle with the external NVIDIA PC (192.168.131.5).

Visualization

RViz Visualization

To view the Dingo current state:

```
roslaunch mmp_viz view_robot.launch
```

This package is useful when checking and integrating depth camera or other auxiliary sensors such as LiDAR sensors.

Rigs

Hardware Configurations

The Dingo-D (Noetic) can be configured with different sensor and accessory rigs.

Standard Configuration

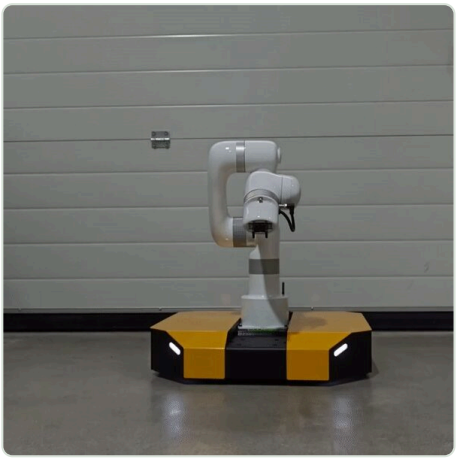
- Clearpath Dingo-D differential base
- Ufactory Lite6 arm
- Hokuyo LiDAR (192.168.131.20)
- Intel RealSense D405 (wrist-mounted)

Network Devices

Device	Network Address	Users	Password
Dingo-D	192.168.131.1	administrator	mybotshop
Hokuyo	192.168.131.20	n/a	n/a
Lite6	192.168.131.50	n/a	n/a
Lite6 Web	192.168.131.50:18333	n/a	n/a

Manipulation

Lite6 MoveIt



Lite6 MoveIt

> **Note:** As the Dingo-D Lite6 is a multi-robot system consisting of Lite6 and Lite6 gripper, the Lite6 Guides are applicable.

For the Lite6 ROS drivers please refer to the ufactory github section. To aid in manipulation tasks please refer to the examples in the mmp_bringup/scripts/mmp_example.py . For recommended manipulation please refer to MoveIt documentation.

Lite6 Utilities

To operate the Lite6 from other means than ROS , the Python and C++ SDKs are available in the utils folder. Direct control of the Lite6 is available via the app. To access the app, navigate into the /utils/app folder and run:

```
./UfactoryStudio-client-linux-1.0.1.AppImage
```

Alternatively, you can install the app image directly from their website and grant root permission to the image.

Connect to the IP address of 192.168.131.50 and you should have complete access to the arm. The app has a built-in plugin to interface with robotiq gripper available in the control panels.

> **Important:** The app can also be accessed via ssh or when connected to the same network as the Lite6 at 192.168.131.50:18333 .

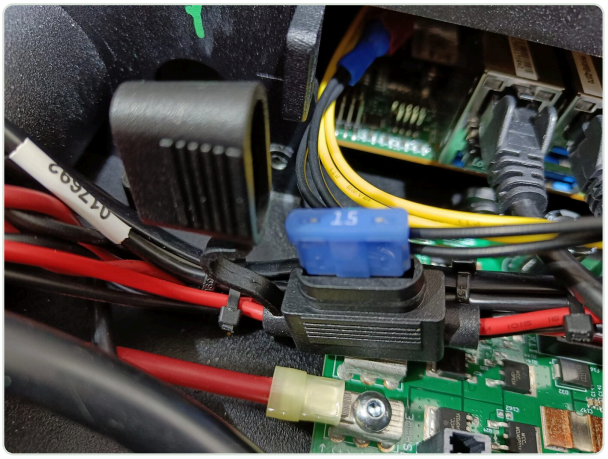
Complete steps for opening the app:

```
ssh -X administrator@192.168.131.1
mybotshop
cd
```

```
cd ros_ws/utils/app
./UfactoryStudio-client-linux-1.0.1.AppImage
```

This can be used when the Lite6 goes to error state. Alternatively, the error can be cleared via the Lite6 python API.

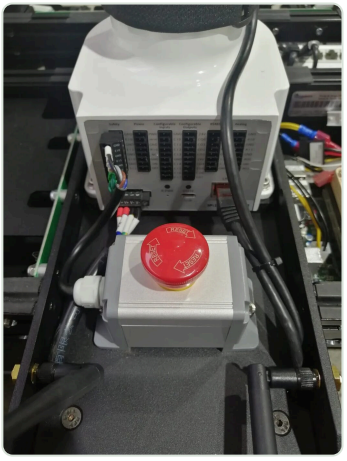
Lite6 Power Fuse



Lite6 Power Fuse

The Lite6 is powered by Dingo-D's power. For safety reasons a 15 Amp fuse is installed. In case of a power surge, the fuse will trigger and will have to be replaced. To replace it, open the Dingo-D's central hatch located under the Lite6, open the rubber cover over the MCU and replace the fuse.

Lite6 E-Stop



Lite6 E-Stop

There's a red Emergency Stop button positioned at the rear of the mounted arm. Pressing this button cuts off the power supply to the arm, providing a manual method to halt the arm in case of any malfunction or emergency situation.

Navigation

Navigation

Navigation packages for ROS Noetic are available through the standard ROS navigation stack.

For SLAM mapping, use:

```
roslaunch mmp_navigation slam.launch
```

For autonomous navigation with a pre-built map:

```
roslaunch mmp_navigation navigation.launch
```

TEB Local Planner

For TEB local planner configuration, refer to:

- TEB Local Planner Guide 1
- TEB Local Planner Guide 2

Sensors

Hokuyo LiDAR

Hokuyo LiDAR

You can start up the Hokuyo driver located at the robot's front:

```
roslaunch mmp_lidars lidars.launch
```

Once running, you should be able to view the laser scan data created by Hokuyo on RViz.

Intel RealSense D405

Intel RealSense D405

To initiate the hardware driver for the RealSense camera mounted on the wrist of the Lite6:

```
roslaunch mmp_realsense wrist_camera.launch
```

If the camera drivers operate correctly, you should be able to view the camera stream on RViz.

Simulation

Gazebo Simulation

The Dingo-D can be simulated in Gazebo Classic.

Launch Simulation:

```
roslaunch mmp_gazebo simulation.launch
```

This launches:

- Gazebo with the Dingo model
- Robot state publisher
- Joint state publisher

Packages

Dingo Base Driver

Dingo Base Driver

To start the hardware-dingo drivers:

```
roslaunch dingo_base base.launch
```

Important: You need not manually execute this command, as the dingo hardware drivers are automatically launched during the robot's startup process.

Lite6 Driver

To start the hardware-lite6 drivers:

```
roslaunch mmp_bringup lite6_driver.launch
```

This driver does not launch on startup. It is required for controlling the Lite6 arm.

Warning: Non-technical persons are not permitted to use the device as the damage it can cause is severe.

Visualization Package

```
roslaunch mmp_viz view_robot.launch
```

LiDAR Package

```
roslaunch mmp_lidars lidars.launch
```

RealSense Package

```
roslaunch mmp_realsense wrist_camera.launch
```

Debugging

Common Issues

Robot not responding to commands:

1. Check if ROS master is running: roscore
2. Verify network connection: ping 192.168.131.1
3. Check ROS topics: rostopic list

No sensor data:

1. Verify sensor connections
2. Check sensor-specific topics
3. Review sensor launch files

Diagnostic Commands

Check ROS Topics:

```
rostopic list
rostopic echo /odom
```

Check ROS Nodes:

```
rosnode list
rosnode info /dingo_base
```

Check TF Tree:

```
rosrun tf view_frames
```

Check Service Status:

```
sudo service {service_name} status
```

Rebuild Workspace

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

Miscellanious

SSH Access

Connect to the robot via SSH:

```
ssh -X administrator@192.168.131.1
# Password: mybotshop
```

Programming Examples

Examples of programming the Lite6 and Dingo-D are provided in:

```
mmp_bringup/scripts/mmp_example.py
```

Message-of-the-Day (MOTD)

To configure a custom MOTD:

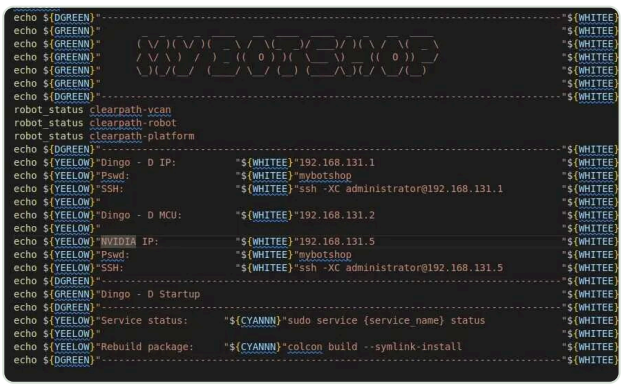
```
sudo chmod -x /etc/update-motd.d/*
sudo nano /etc/update-motd.d/45-mbs-motd
sudo chmod +x /etc/update-motd.d/45-mbs-motd
```

Example MOTD script:

```
#!/bin/sh

robot_status clearpath-vcan
robot_status clearpath-robot
robot_status clearpath-platform

echo "------------------------------------------------------------------------------------"
echo "Dingo - D IP:          192.168.131.1"
echo "Password:              mybotshop"
echo "SSH:                   ssh -XC administrator@192.168.131.1"
echo
echo "Dingo - D MCU:         192.168.131.2"
echo
echo "NVIDIA IP:             192.168.131.5"
echo "Password:              mybotshop"
echo "SSH:                   ssh -XC administrator@192.168.131.5"
echo "------------------------------------------------------------------------------------"
echo "Dingo - D Startup"
echo "------------------------------------------------------------------------------------"
echo "Service status:        sudo service {service_name} status"
echo "Rebuild package:       colcon build --symlink-install"
echo "------------------------------------------------------------------------------------"
```



Dingod Motd

Installation

ROS Noetic Installation

ROS Noetic Installation

The ROS driver for the robots is initially supported for ROS Noetic. All ROS related software should be installed on the Remote-PC . The Dingo-D is pre-installed with the required drivers and ROS distribution.

PC Setup

ROS Noetic installation requires Ubuntu 20.04.

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

ROS Noetic Installation

You may run the following commands to install ROS Noetic or follow the instructions from the ROS wiki.

1. Enter the Ubuntu terminal:

```
cd
```

1. Set 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. Set up the keys:

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

1. Install ROS Noetic:

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

1. Add ROS environment to your bash file:

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

1. Install python dependencies and initialize rosdep:

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

Catkin Workspace

1. Ensure ROS Noetic is correctly setup:

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

1. Install catkin_tools:

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

1. Create your workspace:

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

1. Enable the workspace:

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

1. Add to bashrc to avoid typing every time:

```
gedit ~/.bashrc
```

Add the following line at the end:

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

1. Verify everything is working:

```
echo $ROS_PACKAGE_PATH
```

14 Getting Help & RMA

RMA / Support

How to Raise a Return Merchandise Authorization (RMA) Case

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

Step 1: Open a Topic in Our Forum

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

Step 2: Contact Support for RMA Number

If the issue cannot be resolved remotely through our forum:

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

Important Shipping Information

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

Help Resources

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

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