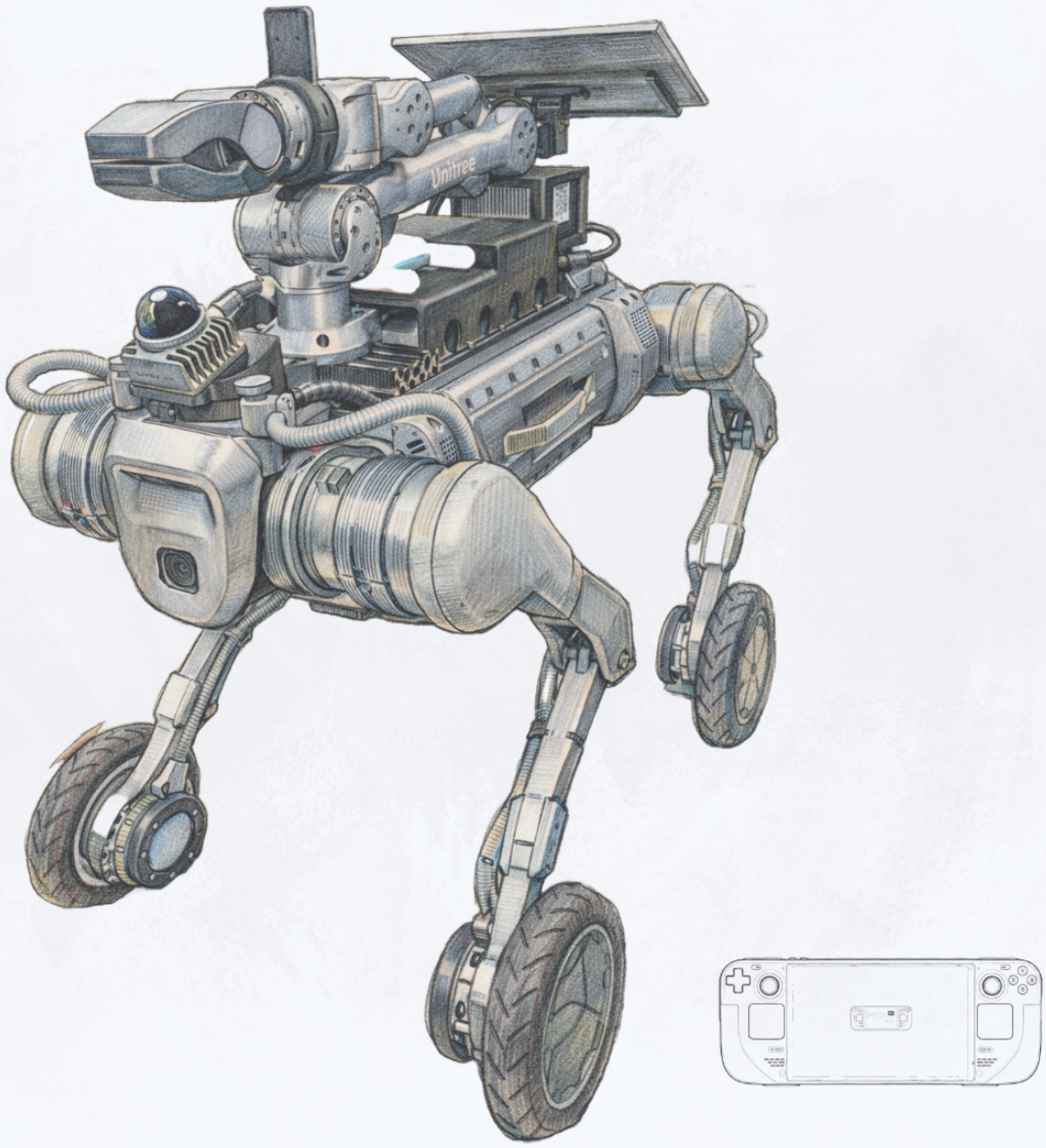
MYBOTSHOP GMBH

QRE Advanced B2-W
Manipulator



MYBOT SHOP

info@mybotshop.de
support@mybotshop.de
forum.mybotshop.de
docs.mybotshop.de

QUADRUPED ROBOTICS

info@quadruped.de
support@quadruped.de
forum.mybotshop.de
docs.quadruped.de

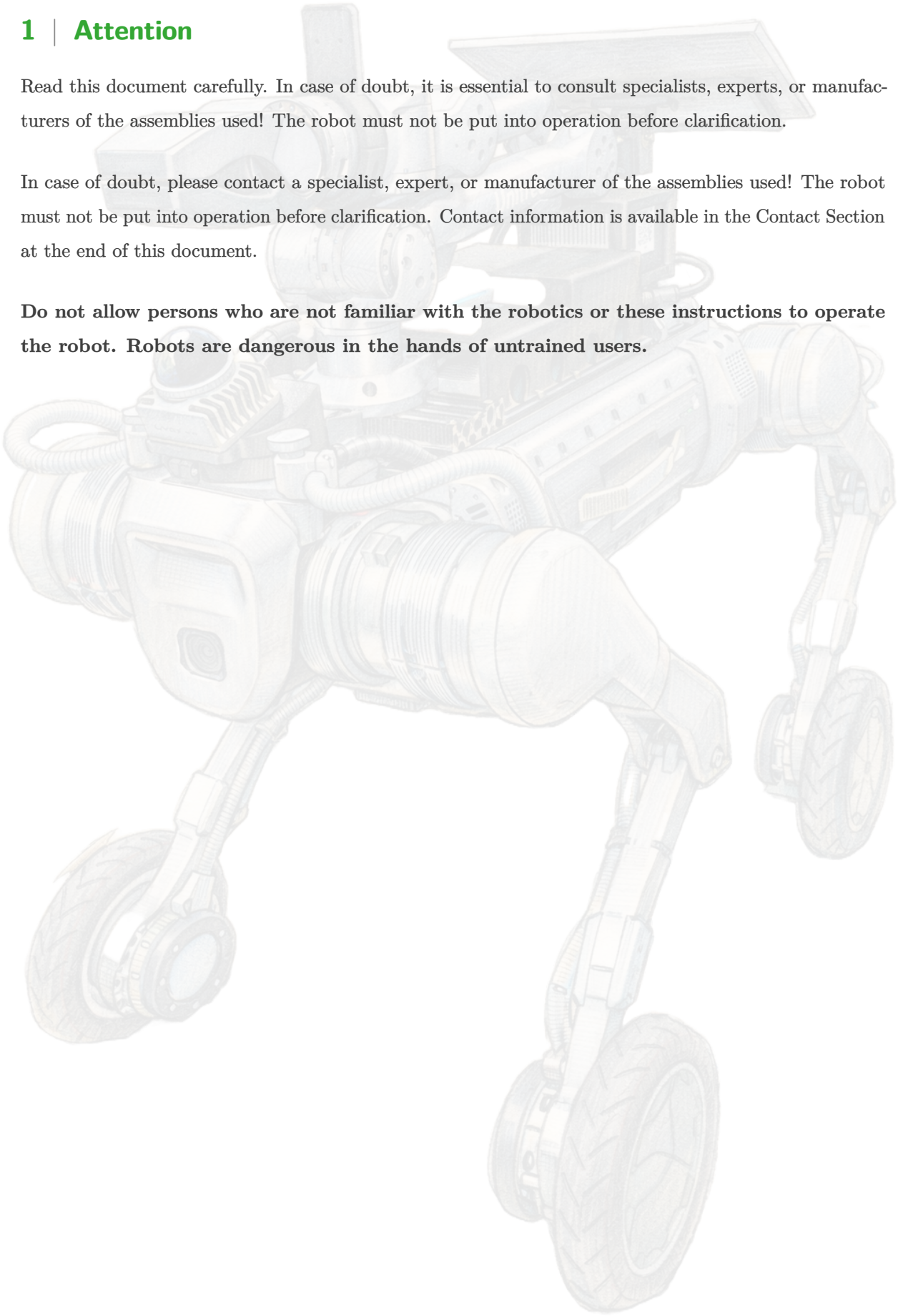
Contents

1 | Attention

Read this document carefully. In case of doubt, it is essential to consult specialists, experts, or manufacturers of the assemblies used! The robot must not be put into operation before clarification.

In case of doubt, please contact a specialist, expert, or manufacturer of the assemblies used! The robot must not be put into operation before clarification. Contact information is available in the Contact Section at the end of this document.

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



2 | Disclaimer

The provided robot, sold by *MYBOTSHOP GmbH*, is an R&D device and does not have CE Marking and/or Certificate of Incorporation. Basic ROS understanding is required. If you are not familiar with ROS, we recommend checking the ROS2 Wiki first.

Please note that the quoted robot is considered a partly completed machine under the *Machinery Directive2006/42/EC*, and it does not carry a CE marking.

The client acknowledges and agrees that any information or materials provided by *MYBOTSHOP GmbH* are for R&D 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.

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

3 | Safety and Robot Control

WARNING: The Unitree B2W is a powerful robot capable of moving at high speeds. In the event of a network disconnection or software failure, the robot may continue moving uncontrollably and cause damage to itself, equipment, or personnel. **Always keep the Unitree remote controller on hand during operation for emergency stops!**

3.1 | Robot Mode Control

The B2W supports various operational modes controlled via ROS 2 services.

3.1.1 | Available Modes

Mode	Description
`damp`	Damping mode (safe state)
`stand_up`	Stand up from lying position
`stand_down`	Lie down safely
`recovery`	Recovery mode for fault conditions
`stop_move`	Stop all movement
`gait_idle`	Idle gait (standing)
`gait_trot`	Trotting gait
`gait_trot_running`	Running trot gait
`gait_visualwalk`	Visual walking mode
`gait_flatwalk`	Flat terrain walking

3.1.2 | Mode Control Examples

```
# Stand up
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'stand_up'}"

# Stand down (safe shutdown position)
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'stand_down'}"

# Enter damping mode
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'damp'}"
```

```
# Set trotting gait
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'gait_trot'}"

# Recovery mode
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'recovery'}"
```

3.2 | Speed and Body Height Control

3.2.1 | Speed Settings

```
# Set low speed
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'speed_low'}"

# Set high speed
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'speed_high'}"
```

3.2.2 | Body Height Settings

```
# Low body height
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'body_height_low'}"

# Medium body height
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'body_height_mid'}"

# High body height
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'body_height_high'}"
```

3.3 | Z1 Arm Control

EXPERIMENTAL: The Z1 driver is under active development.

If your B2W is equipped with the Unitree Z1 arm:

```
# Enable the arm
ros2 service call /b2_unit_001/z1/enable std_srvs/srv/Trigger

# Send trajectory command
ros2 topic pub --once /b2_unit_001/z1/joint_trajectory \
  trajectory_msgs/msg/JointTrajectory "{
  joint_names: ['joint1', 'joint2', 'joint3',
    'joint4', 'joint5', 'joint6', 'jointGripper'],
  points: [{
    positions: [0.0, 1.57, -1.57, 0.0, 0.0, 0.0, 0.0],
    time_from_start: {sec: 3, nanosec: 0}
  }]
}"

# Disable the arm
ros2 service call /b2_unit_001/z1/disable std_srvs/srv/Trigger
```

4 | Quick Start

Important: This manual is for the Unitree B2W (Wheeled Quadruped Robot Platform). For official Unitree documentation, refer to the Unitree B2 Developer Guide.

Documentation Links:

- MYBOTSHOP B2 Documentation
- Unitree B2 Developer Guide
- Unitree Z1 Arm Parameters

5 | Quick Start ROS2

5.1 | Power On

1. Release the emergency stop button by turning it clockwise.
2. Press the power button to turn on the robot.
3. Wait for the robot to boot up completely (approximately 1-2 minutes).
4. The B2 services will start automatically.
5. Verify the webserver is accessible at 192.168.123.164:9000.

5.2 | Power Off

Important: Always follow the proper shutdown procedure to prevent data corruption and hardware damage.

1. **Stand down the robot**:

```
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'stand_down'}"
```

2. **Engage the emergency stop** by pressing it down.
3. **Shutdown the robot** via SSH or Cockpit terminal:

```
sudo shutdown now
```

4. **Turn off the main power switch**.

Note: For a quick reboot without full power cycle, use `sudo reboot` instead of `sudo shutdown now`.

5.3 | Network Connection

1. Connect to the robot's network via Ethernet.
2. Configure a static IP address on your host PC: **192.168.123.51** with netmask **24**.
3. The Nvidia Module IP is **192.168.123.165**.
4. Access the webserver at **192.168.123.164:9000** (HTTPS).
5. The webserver username is **admin** and password is **mybotshop**.

5.4 | Teleoperation

Teleoperate using the keyboard teleop package:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  cmd_vel:=/b2_unit_001/hardware/cmd_vel
```

Note: Replace `b2_unit_001` with your robot's namespace. Reduce speed by pressing **z** until **0.1** for safe indoor operation.

5.5 | Visualization

View the robot state in RViz2:

```
ros2 launch b2_viz view_robot.launch.py
```

6 | Quick Start Dev

6.1 | Network Table

IP Address	Device	Username	Password
192.168.123.161	B2 MCU	-	-
192.168.123.162	B2 Robosense Lidar	-	-
192.168.123.164	B2 Onboard PC	unitree	Unitree0408
192.168.123.165	Nvidia Module	administrator	mybotshop
192.168.123.110	Unitree Z1 Arm	-	-
192.168.123.120	Livox Mid360 (Front)	-	-
192.168.123.121	Livox Mid360 (Rear)	-	-
192.168.123.170	Video Stream (Secondary)	-	-
192.168.123.150	B2 Steamdeck	deck	mybotshop
192.168.123.200	B2 Access Point (5G/2.4G)	B2-Unit-5G	mybotshop
192.168.123.164:9000	Webserver	admin	mybotshop
192.168.123.164:9090	ROS Bridge	-	-

6.2 | SSH Connection

Access the robot via SSH:

```
# Connect to Nvidia Module
ssh administrator@192.168.123.165

# Connect to B2 Onboard PC
ssh unitree@192.168.123.164
```

6.3 | Check Service Status

```
sudo systemctl status b2-webserver
sudo systemctl status b2-hardware
```

6.4 | Webserver

The webserver module comes pre-installed and should be accessible directly at https://192.168.123.164:9000/ or via the WiFi IP to which the robot is connected.

6.4.1 | Login

- Username: **admin**
- Password: **mybotshop**

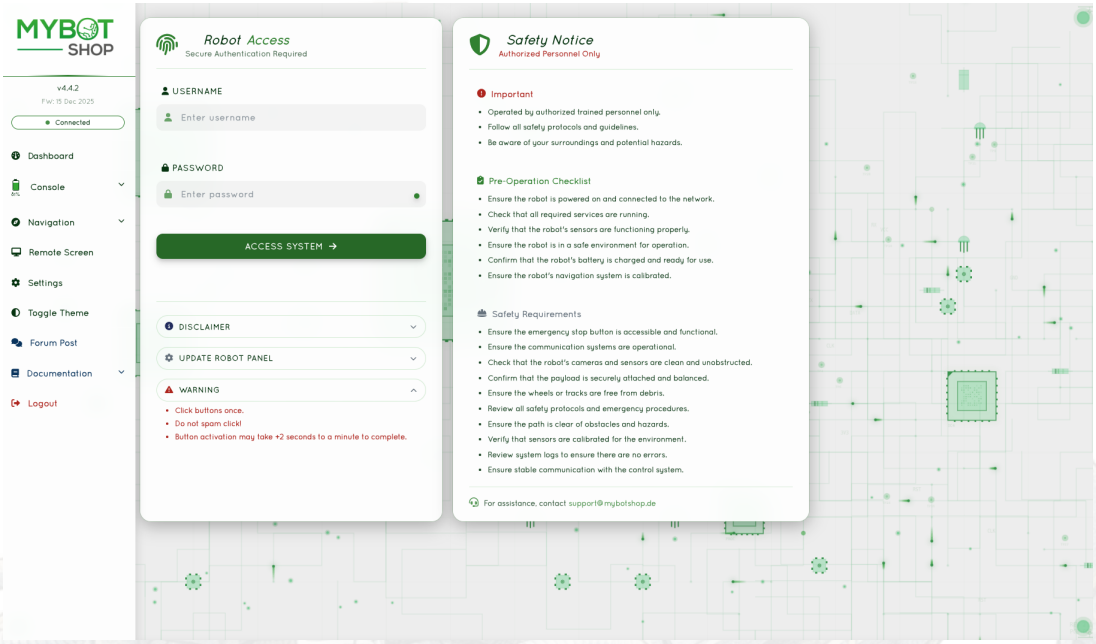


Figure 6.1: Webserver Login Page

6.4.2 | Features

- HTTP/HTTPS server with authentication
- VNC desktop streaming (1920x1080)
- ROS bridge WebSocket interface
- GPS waypoint management
- Service status monitoring
- Rosbag recording
- Dynamixel servo control sliders
- AI/LLM integration (optional)

6.4.3 | Dashboard

- View IP Address of the B2W
- View System load
- Enable/Disable the B2W ROS2 Services
- Record System logs
- Battery status monitoring
- Pre-configured action buttons

- Web Joystick for teleoperation
- Online video stream (front and rear cameras)

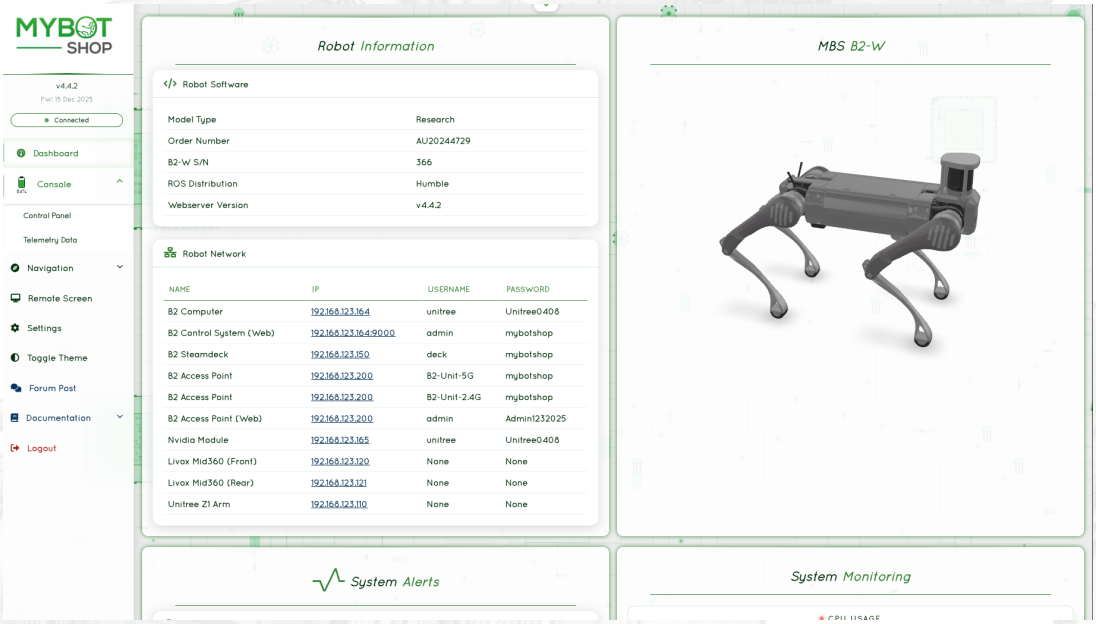


Figure 6.2: Webserver Dashboard

6.4.4 | Remote Desktop (VNC)

- Access on-board screen of the B2W's computer
- VNC password: **mybotshop**

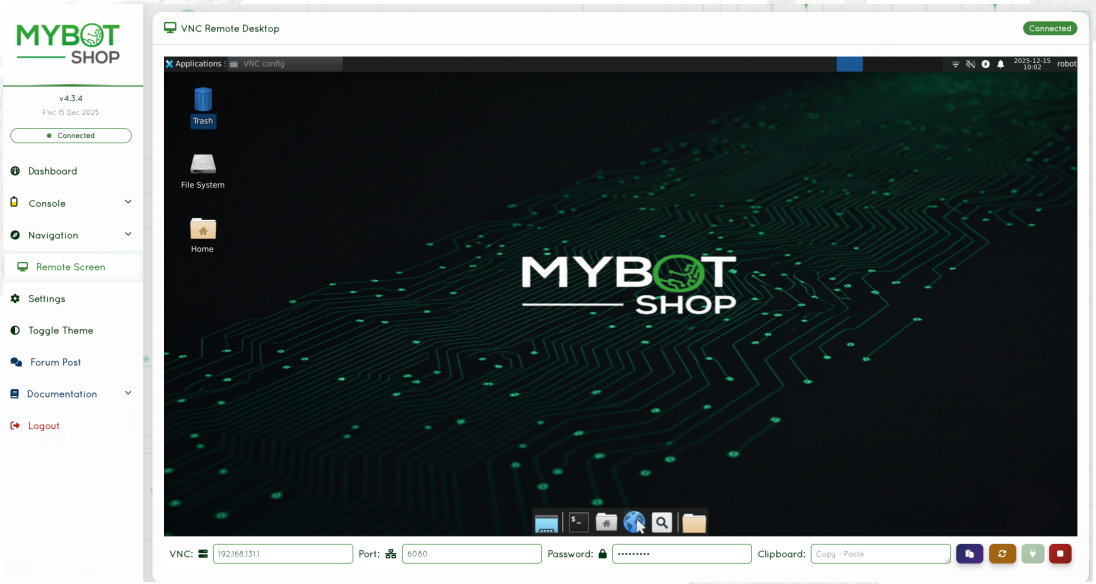


Figure 6.3: Remote Desktop (VNC) Interface

6.4.5 | System

- System information and diagnostics
- Service management

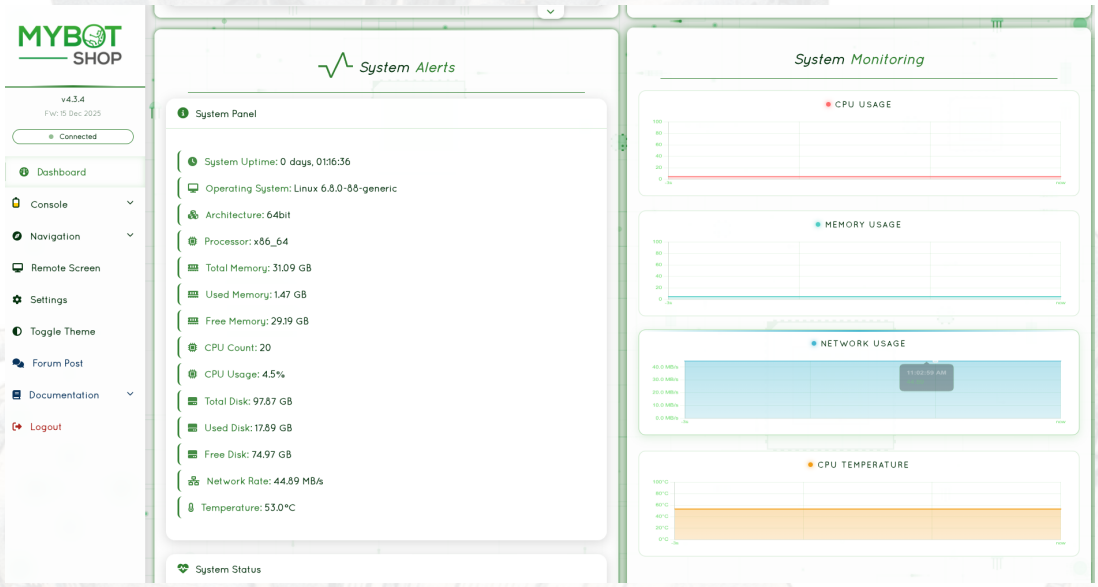


Figure 6.4: System Information Page

6.4.6 | Console

- Web-based terminal access
- Command execution interface

6.4.7 | Navigation Indoor

- Map visualization and navigation control

6.4.8 | Navigation Outdoor

- GPS-based outdoor navigation
- Waypoint management

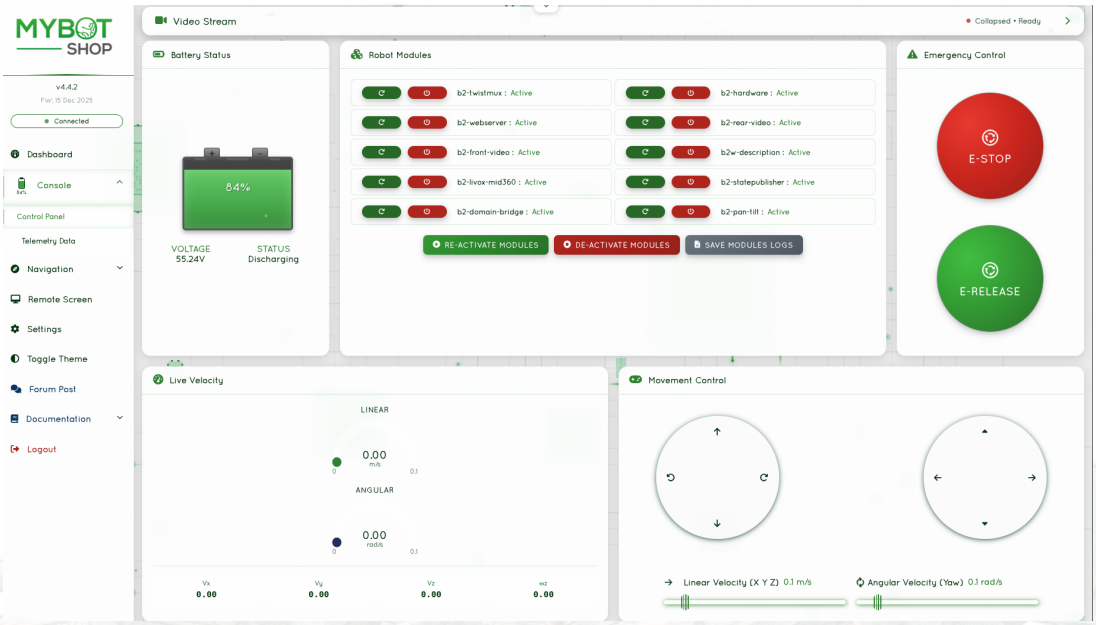


Figure 6.5: Web Console Interface

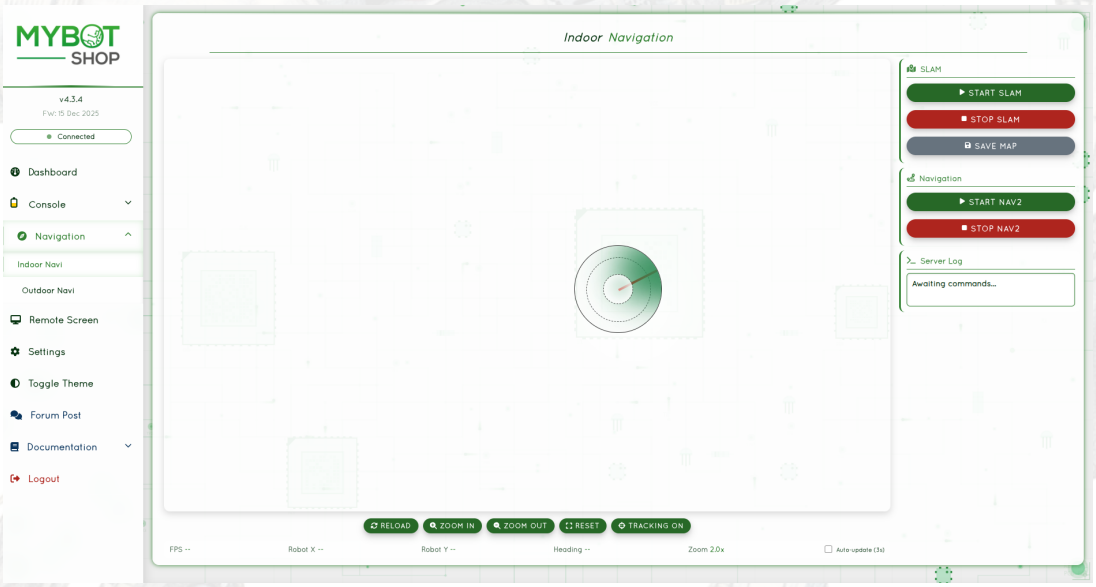


Figure 6.6: Indoor Navigation Interface

6.5 | Robot Modes

Available modes via ROS 2 service:

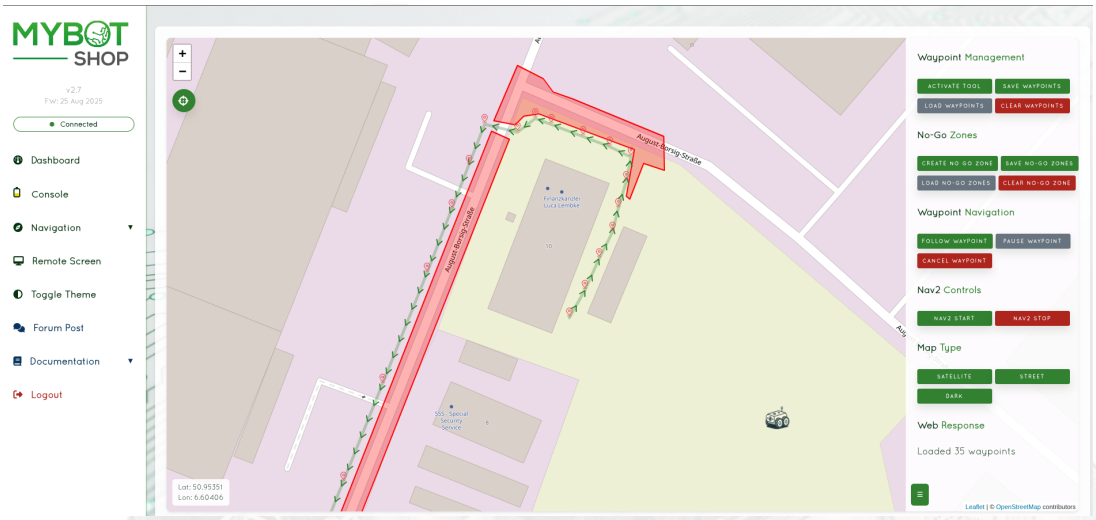


Figure 6.7: Outdoor Navigation Interface

Mode	Description
`damp`	Damping mode
`stand_up`	Stand up from lying
`stand_down`	Lie down
`recovery`	Recovery mode
`stop_move`	Stop movement
`gait_idle`	Idle gait
`gait_trot`	Trotting gait
`gait_trot_running`	Running trot
`gait_visualwalk`	Visual walking
`gait_flatwalk`	Flat terrain walk
`speed_low` / `speed_high`	Speed settings
`body_height_low/mid/high`	Body height

6.5.1 | Mode Control Examples

```
# Stand up
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'stand_up'}"

# Stand down
ros2 service call /b2_unit_001/hardware/modes \
  b2_interface/srv/B2Modes "{request_data: 'stand_down'}"

# Set trotting gait
ros2 service call /b2_unit_001/hardware/modes \
```

```
b2_interface/srv/B2Modes "{request_data: 'gait_trot'}"
```

7 | ROS2 Modules

7.1 | Overview

The QRE B2W platform provides a comprehensive ROS 2 software stack for the Unitree B2 wheeled quadruped robot, including:

- Hardware interface with Unitree SDK 2.0
- Multi-sensor integration (LiDAR, cameras, GPS, IMU)
- Pan-tilt servo control via Dynamixel
- Nav2 navigation stack with SLAM
- Web-based remote operation interface
- Gazebo Fortress simulation environment

ROS Domain ID: All packages use `ROS_DOMAIN_ID=10` for inter-process communication.
Default Namespace: `b2_unit_001` (configurable via `B2_NS` environment variable)
To launch the B2W RViz visualization:

```
ros2 launch b2_viz view_robot.launch.py
```

7.2 | Tele-Operation

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  cmd_vel:=/b2_unit_001/hardware/cmd_vel
```

7.3 | Auto Drivers Startup

The B2W uses robot_upstart services for automatic startup. To check service status:

```
sudo systemctl status b2-hardware
sudo systemctl status b2-webserver
```

To restart a service:

```
sudo systemctl restart b2-hardware
```

7.4 | Environment Variables

The following environment variables are used by the B2W system:

Variable	Default	Description
B2_NS	b2_unit_001	Robot namespace
ROS_DOMAIN_ID	10	ROS 2 Domain ID
B2_DESC	1	Enable B2 description
B2_Z1	0	Enable Z1 arm in URDF

Note: Replace `b2_unit_001` with your robot's namespace.

8 | Sensors

8.1 | LiDAR - Livox Mid360

The B2W is equipped with dual Livox Mid360 LiDAR sensors for 3D perception.

8.1.1 | Network Configuration

Sensor	IP Address
Livox Mid360 (Front)	192.168.123.120
Livox Mid360 (Rear)	192.168.123.121

8.1.2 | Launch Options

```
# Launch both LiDARs
ros2 launch b2_lidars livox_mid360.launch.py lidar_position:=both

# Launch front only
ros2 launch b2_lidars livox_mid360.launch.py lidar_position:=front

# Launch rear only
ros2 launch b2_lidars livox_mid360.launch.py lidar_position:=rear
```

8.1.3 | Launch Arguments

Argument	Default	Description
`lidar_position`	both	front, rear, or both
`xfer_format`	0	Data transfer format
`publish_freq`	10.0	Publishing frequency (Hz)
`namespace`	-	Topic namespace

8.1.4 | Published Topics

Topic	Type
`livox/front/lidar`	`sensor_msgs/PointCloud2`
`livox/rear/lidar`	`sensor_msgs/PointCloud2`
`livox/front/imu`	`sensor_msgs/Imu`
`livox/rear/imu`	`sensor_msgs/Imu`
`scan`	`sensor_msgs/LaserScan`

8.2 | Depth Camera - RealSense D435i

The B2W can be equipped with Intel RealSense D435i depth cameras.

8.2.1 | Documentation

For RealSense documentation, see: https://github.com/IntelRealSense/realsense-ros

8.3 | Pan-Tilt System (Dynamixel)

The `b2_dynamixel` package controls Dynamixel servos for pan-tilt mechanisms using Protocol 2.0.

8.3.1 | Hardware Setup

- **Supported Servos:** XM, XH, XC, XL430, XW series
- **Interface:** U2D2 USB adapter
- **Serial Port:** `/dev/b2/dynamixel`
- **Baud Rate:** 57600

8.3.2 | Configuration

Configuration file: `b2_dynamixel/config/dynamixel.yaml`

```
dynamixel_driver:
  ros__parameters:
    port: '/dev/b2/dynamixel'
    baudrate: 57600
    servo_ids: [0, 1]
    # Position limits (raw Dynamixel units: 0-4095)
    position_min: [1150, 230]
    position_max: [3200, 900]
    # Motion profile
    profile_velocity: 100
    profile_acceleration: 50
    # Publishing
    publish_rate: 20.0
    position_topic_prefix: 'dynamixel/servo'
    feedback_topic: 'dynamixel/joint_states'
```

8.3.3 | Topics and Services

Topic/Service	Type	Description
dynamixel/servo_0/command	Float64	Pan (1150-3200)
dynamixel/servo_1/command	Float64	Tilt (230-900)
dynamixel/joint_states	JointState	Positions (rad)
dynamixel/reboot	Trigger	Clear errors

8.3.4 | Usage

```
# Launch driver
ros2 launch b2_dynamixel dynamixel.launch.py

# Center pan servo (default: 2170)
ros2 topic pub --once /dynamixel/servo_0/command \
  std_msgs/msg/Float64 "data: 2170"

# Center tilt servo (default: 565)
ros2 topic pub --once /dynamixel/servo_1/command \
  std_msgs/msg/Float64 "data: 565"

# Read joint states
ros2 topic echo /dynamixel/joint_states
```

```
# Reboot servos (clear hardware errors)
ros2 service call /dynamixel/reboot std_srvs/srv/Trigger
```

8.3.5 | Webserver Integration

The servos are controllable via the web interface sliders:

- **Servo 0 (Pan):** Range 1150-3200, Default 2170
- **Servo 1 (Tilt):** Range 230-900, Default 565

Joint states are published to `/dynamixel/joint_states` in radians for URDF visualization.

8.3.6 | Udev Rules

To create persistent device symlinks:

```
echo 'SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", \
  ATTRS{idProduct}=="6014", SYMLINK+="b2/dynamixel", \
  MODE="0666"' | sudo tee /etc/udev/rules.d/99-dynamixel.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

8.4 | Z1 Robotic Arm

EXPERIMENTAL: The Z1 driver is under active development.

The `b2_z1` package provides a ROS 2 driver for the Unitree Z1 robotic arm.

8.4.1 | Quick Start

```
# Launch the driver
ros2 launch b2_z1 z1.launch.py

# Enable the arm (required before sending commands)
ros2 service call /b2_unit_001/z1/enable std_srvs/srv/Trigger

# Send trajectory command
ros2 topic pub --once /b2_unit_001/z1/joint_trajectory \
  trajectory_msgs/msg/JointTrajectory "{
  joint_names: ['joint1', 'joint2', 'joint3',
    'joint4', 'joint5', 'joint6', 'jointGripper'],
```

```
  points: [{
    positions: [0.0, 1.57, -1.57, 0.0, 0.0, 0.0, 0.0],
    time_from_start: {sec: 3, nanosec: 0}
  }]
}"

# Disable the arm
ros2 service call /b2_unit_001/z1/disable std_srvs/srv/Trigger
```

8.4.2 | Topics and Services

Name	Type	Description
z1/joint_states	sensor_msgs/JointState	Joint positions
z1/joint_trajectory	trajectory_msgs/JointTrajectory	Commands
z1/enable	std_srvs/Trigger	Enable arm
z1/disable	std_srvs/Trigger	Disable arm

8.4.3 | URDF Integration

Enable Z1 in the robot description:

```
export B2_Z1=1
ros2 launch b2_description display.launch.py
```

9 | Simulation

The B2W simulation package uses ROS2 Humble with Gazebo Fortress, featuring the complete B2W platform with wheeled quadruped locomotion.

9.1 | Prerequisites

- ROS 2 Humble
- Gazebo Fortress
- gz_ros2_control

9.2 | B2 Simulation (Legged)

```
ros2 launch b2_gazebo b2_fortress_simulation.launch.py
```

9.3 | B2W Simulation (Wheeled)

```
ros2 launch b2_gazebo b2w_fortress_simulation.launch.py
```

To kill Gazebo zombie processes:

```
ros2 run b2_gazebo kill_gz.sh
```

9.4 | Environment Variables

Configure the simulation using environment variables before launch:

Variable	Default	Description
B2_DESC	1	Enable B2 robot description
B2_Z1	0	Include Z1 robotic arm

9.5 | RViz2 Visualization

To visualize the robot in RViz2:

```
ros2 launch b2_viz view_robot.launch.py
```

9.6 | Teleoperation (Keyboard)

To teleoperate the B2W using your keyboard:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  cmd_vel:=/b2_unit_001/hardware/cmd_vel
```

Note: Reduce speed by pressing **z** until **0.1** for safe indoor operation.

9.7 | Joint Trajectory Control (B2 Legged)

```
ros2 action send_goal \
  /joint_effort_controller/follow_joint_trajectory \
  control_msgs/action/FollowJointTrajectory -f "{
  trajectory: {
    joint_names: [
      'FL_hip_joint', 'FL_thigh_joint', 'FL_calf_joint',
      'FR_hip_joint', 'FR_thigh_joint', 'FR_calf_joint',
      'RL_hip_joint', 'RL_thigh_joint', 'RL_calf_joint',
      'RR_hip_joint', 'RR_thigh_joint', 'RR_calf_joint'
```

```
    ],
    points: [{
      positions: [0.0, 0.9, -1.5, 0.0, 0.9, -1.5,
        0.0, 0.9, -1.5, 0.0, 0.9, -1.5],
      time_from_start: {sec: 2, nanosec: 0}
    }]
  }
}"
```

9.8 | B2W Wheel Velocity Control

```
# Forward
ros2 topic pub /velocity_controller/commands \
  std_msgs/msg/Float64MultiArray \
  "{data: [-1.0, -1.0, -1.0, -1.0]}"

# Backward
ros2 topic pub /velocity_controller/commands \
  std_msgs/msg/Float64MultiArray \
  "{data: [1.0, 1.0, 1.0, 1.0]}"

# Stop
ros2 topic pub /velocity_controller/commands \
  std_msgs/msg/Float64MultiArray \
  "{data: [0.0, 0.0, 0.0, 0.0]}"
```

9.9 | Robot Description

9.9.1 | URDF Variants

```
# Standard B2 (legged)
export B2_DESC=1
ros2 launch b2_description b2_description.launch.py

# Wheeled B2W
ros2 launch b2_description b2w_description.launch.py
```

9.9.2 | Convert Xacro to URDF

```
export B2_DESC=1
ros2 run xacro xacro \
  /opt/mybotshop/src/mybotshop/b2_description/xacro/robot.xacro \
  > /opt/mybotshop/src/mybotshop/b2_description/xacro/b2.urdf
```

9.10 | Simulation Troubleshooting

If controllers are not starting, manually activate them:

```
ros2 control set_controller_state \
  -c /b2_unit_001/controller_manager \
  joint_state_broadcaster configure

ros2 control set_controller_state \
  -c /b2_unit_001/controller_manager \
  joint_state_broadcaster activate
```

Check hardware interfaces:

```
ros2 control list_hardware_interfaces \
  -c /b2_unit_001/controller_manager
```

10 | Navigation

The B2W uses the Nav2 navigation stack for autonomous navigation.

10.1 | SLAM (2D Mapping)

Run SLAM to create a 2D occupancy grid map using the LiDAR sensor:

```
ros2 launch b2_nav2 slam.launch.py
```

Drive the robot slowly (recommended 0.2 m/s) using the joystick controller or keyboard teleop to build the map:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  cmd_vel:=/b2_unit_001/controls/cmd_vel
```

Once satisfied with the map, save it using:

```
ros2 run nav2_map_server map_saver_cli -f \
  /opt/mybotshop/src/mybotshop/b2_nav2/maps/custom_map \
  --ros-args --remap map:=/b2_unit_001/map
```

> **Note:** Replace `b2_unit_001` with your robot's namespace. Maps are saved to the `b2_nav2/maps/` directory. After saving, rebuild the workspace with `colcon build --symlink-install` so the new map can be found.

10.1.1 | SLAM Configuration

Configuration file: `b2_nav2/param/nav2_slam.yaml`

```
slam_toolbox:
  ros__parameters:
    solver_plugin: solver_plugins::CeresSolver
    resolution: 0.05
    max_laser_range: 40.0
    mode: mapping
```

10.2 | Odom Navigation

Run odometry-based navigation without a pre-built map. This mode navigates in the global `map` frame using only odometry for localization:

```
ros2 launch b2_nav2 odom_navi.launch.py
```

This is useful for relative navigation where you don't have a map of the environment yet.

10.3 | Map-Based Navigation

Use a pre-built 2D map for autonomous navigation with AMCL localization:

```
ros2 launch b2_nav2 map_navi.launch.py
```

This launches the full Nav2 stack including:

- Map Server - loads pre-built occupancy grid maps
- AMCL - Adaptive Monte Carlo Localization
- Controller Server - path following (DWB controller)
- Planner Server - global path planning (NavFn)
- Behavior Server - recovery behaviors
- Waypoint Follower - for multi-point navigation

10.4 | GPS-Based Navigation

For outdoor GPS-based navigation:

```
ros2 launch b2_nav2 gps_navi.launch.py
```

11 | ROS2 Package Reference

Detailed documentation for each package in the QRE B2W workspace.

11.1 | Workspace Structure

The QRE B2W workspace is organized into two main categories: **mybotshop** (custom MBS packages) and **third_party** (external dependencies).

11.1.1 | Core Packages

Package	Version	Description
`b2_platform`	1.0.0	Unitree SDK hardware interface, odometry, video streaming
`b2_interface`	1.0.0	Custom ROS 2 services (B2Modes.srv, B2Light.srv)
`b2_dynamixel`	1.0.0	Dynamixel servo control for pan-tilt mechanism
`b2_bringup`	1.0.0	System startup and service installer
`b2_description`	1.0.0	URDF/Xacro robot descriptions
`b2_z1`	1.0.0	Unitree Z1 arm driver

11.1.2 | Sensor Packages

Package	Version	Description
b2_lidars	1.2.0	Livox Mid360 dual-LiDAR integration
b2_depth_camera	1.2.0	RealSense D435i depth camera
b2_usbcam	1.2.0	USB camera streaming
b2_gps	1.2.0	GPS/GNSS integration (Fixposition, Drotek)
b2_vision_action	1.0.0	Computer vision and object detection

11.1.3 | Navigation and Control

Package	Version	Description
b2_control	1.2.0	Twist mux, teleop, EKF localization
b2_nav2	1.0.0	Nav2 navigation stack, SLAM
b2_webserver	2.0.0	Web-based remote operation interface
b2_viz	1.0.0	RViz visualization configurations
b2_gazebo	1.0.0	Gazebo Fortress simulation
b2_steamdeck	1.0.0	Steam Deck wireless teleoperation

11.2 | b2_platform

The b2_platform package provides ROS 2 integration with the Unitree SDK 2.0.

11.2.1 | Key Launch Files

- hardware.launch.py - Main hardware control
- state_publisher.launch.py - Robot state publishing
- front_video.launch.py - Front camera streaming
- rear_video.launch.py - Rear camera streaming
- bridge.launch.py - ROS domain bridging

11.2.2 | Configuration

Configuration file: b2_platform/config/b2_platform.yaml

```
robot_odom_topic: "platform/odom"
robot_odom_publisher_on: true
transform_odom_frame: "odom"
transform_base_frame: "base_link"
front_camera_port: 1720
```

```
rear_camera_port: 1721
```

11.3 | b2_description

URDF and Xacro robot description files for the B2W platform.

11.3.1 | Usage

```
# Standard B2
ros2 launch b2_description b2_description.launch.py

# Wheeled B2W
ros2 launch b2_description b2w_description.launch.py
```

11.4 | b2_gazebo

Gazebo Fortress simulation package for the B2W.

11.4.1 | Usage

```
# B2 legged simulation
ros2 launch b2_gazebo b2_fortress_simulation.launch.py

# B2W wheeled simulation
ros2 launch b2_gazebo b2w_fortress_simulation.launch.py
```

11.5 | b2_nav2

Nav2-based navigation stack for the B2W platform.

11.5.1 | Launch Files

Launch File	Description
`slam.launch.py`	2D SLAM mapping mode
`odom_navi.launch.py`	Odometric navigation (no map)
`map_navi.launch.py`	Full autonomous navigation with map
`gps_navi.launch.py`	GPS-based outdoor navigation

11.6 | b2_lidars

LiDAR sensor bringup package for dual Livox Mid360 sensors.

11.6.1 | Key Launch Files

- `livox_mid360.launch.py` - Dual Livox Mid360 LiDAR

11.6.2 | IP Configuration

Sensor	IP Address
Livox Mid360 (Front)	192.168.123.120
Livox Mid360 (Rear)	192.168.123.121

11.7 | b2_viz

RViz2 visualization package with pre-configured displays.

11.7.1 | Usage

```
ros2 launch b2_viz view_robot.launch.py
```

11.8 | b2_webserver

Web-based remote operation interface with HTTP/HTTPS web interface, VNC streaming, ROS bridge, and remote control.

11.8.1 | Features

- Web dashboard at 192.168.123.164:9000
- ROSBridge WebSocket interface at port 9090
- VNC streaming for remote visualization (port 6080)
- Service activation/deactivation controls
- GPS waypoint management
- Rosbag recording
- Dynamixel servo control sliders

11.8.2 | Default Credentials

- Username: **admin**
- Password: **mybotshop**

11.8.3 | Configuration

Configuration file: `b2_webserver/config/robot_webserver.yaml`

```
web_server_ip: "0.0.0.0"
web_server_port: 9000
web_user: "admin"
web_password: "mybotshop"
robot_cmd_vel: "webserver/cmd_vel"
robot_max_linear_velocity: 0.1
robot_max_angular_velocity: 0.1
robot_services:
  - "b2-twistmux"
  - "b2-hardware"
  - "b2-webserver"
  - "b2-rear-video"
  - "b2-front-video"
  - "b2w-description"
  - "b2-livox-mid360"
  - "b2-statepublisher"
  - "b2-domain-bridge"
  - "b2-pan-tilt"
```

11.9 | b2_control

Control system package for robot base. Provides twist mux, teleop configuration, and EKF localization.

11.9.1 | Twist Mux Priority

The twist multiplexer arbitrates between multiple velocity command sources:

Source	Priority	Timeout
E-Stop Lock	255	None
Steamdeck Joy	20	0.5s
Logitech Joy	15	0.5s
Webserver	10	0.5s
Interactive Marker	7	0.5s
External cmd_vel	5	0.5s
Autonomous High	3	0.5s
Autonomous Mid	2	0.5s
Autonomous Low	1	0.5s

11.9.2 | Key Launch Files

- `twistmux.launch.py` - Twist mux configuration
- `control.launch.py` - Full control stack
- `teleop.launch.py` - Teleop configuration
- `ekf_localization.launch.py` - EKF localization

11.9.3 | Teleop Configuration

Logitech F710:

```
ros2 launch b2_control teleop.launch.py
```

PS4 Controller: Configuration file: `b2_control/config/teleop_ps4.yaml`

11.9.4 | EKF Localization

```
ros2 launch b2_control ekf_localization.launch.py
```

Configuration file: `b2_control/config/localization.yaml`

11.10 | b2_interface

Custom ROS2 message and service definitions for B2W packages.

11.10.1 | Key Services

- `B2Modes.srv` - Robot mode control (stand_up, stand_down, gaits, etc.)
- `B2Light.srv` - LED light control

11.11 | b2_z1

ROS 2 driver for the Unitree Z1 robotic arm.

EXPERIMENTAL: This driver is under active development.

11.11.1 | Topics and Services

- `z1/joint_states` - Joint positions, velocities, efforts
- `z1/joint_trajectory` - Trajectory commands
- `z1/enable` - Enable arm control
- `z1/disable` - Disable arm

11.12 | b2_dynamixel

Dynamixel servo control package for pan-tilt mechanisms using Protocol 2.0.

11.12.1 | Configuration

- Serial port: `/dev/b2/dynamixel`
- Baud rate: 57600
- Servo IDs: 0 (Pan), 1 (Tilt)

11.13 | b2_steamdeck

The `b2_steamdeck` package enables wireless teleoperation of the B2W robot using a Steam Deck as a portable controller with audio feedback.

11.13.1 | Steam Deck Setup

1. Install ROS 2 Humble on Steam Deck:

Follow the standard ROS 2 installation for Arch Linux or use a containerized approach.

2. Clone and build the workspace:

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
git clone <repository_url> mybotshop
cd ~/ros2_ws
colcon build --packages-select b2_steamdeck b2_interface
```

```
source install/setup.bash
```

3. Configure sudo permissions for joystick access:

```
sudo EDITOR=nano visudo
```

Add at the bottom:

```
deck ALL=(ALL) NOPASSWD: /usr/bin/python3 \
  /home/deck/ros2_ws/src/mybotshop/b2_steamdeck/b2_steamdeck/libjoypy.py
```

4. Create desktop shortcut:

```
nano ~/Desktop/StartB2Controller.desktop
```

Paste:

```
[Desktop Entry]
Type=Application
Name=B2 Controller
Exec=bash -c "source /opt/ros/humble/setup.bash && \
  source ~/ros2_ws/install/setup.bash && \
  ros2 launch b2_steamdeck system.launch.py"
Icon=utilities-terminal
Terminal=true
```

Make executable:

```
chmod +x ~/Desktop/StartB2Controller.desktop
sudo update-desktop-database
```

11.13.2 | Network Configuration

Connect the Steam Deck to the same network as the B2W robot:

Device	IP Address	Notes
B2 Onboard PC	192.168.123.164	Robot control
Steam Deck	192.168.123.x	Any available IP

WiFi Setup:

1. Connect to the robot's WiFi network or same LAN
2. Ensure `ROS_DOMAIN_ID=10` is set
3. Verify connectivity: `ping 192.168.123.164`

11.13.3 | Controls

Movement (requires L1 held):

Input	Action
L1 + Left Stick	Linear X/Y movement
L1 + Right Stick X	Angular rotation
R1 + Sticks	Turbo mode (faster speeds)

Robot Commands:

Button Combo	Action
L2 + D-Pad Up	Stand up
L2 + D-Pad Down	Sit down

Speed Configuration:

Mode	Linear X	Linear Y	Angular
Normal	0.7 m/s	0.6 m/s	0.8 rad/s
Turbo	1.0 m/s	1.0 m/s	0.8 rad/s

11.13.4 | Launch

On the Steam Deck:

```
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
export ROS_DOMAIN_ID=10
ros2 launch b2_steamdeck system.launch.py
```

Or click the desktop shortcut created during setup.

Audio Feedback:

The Steam Deck provides audio cues for:

- Controller activation
- Robot connection established
- Robot connection lost

Audio can be disabled in the config file:

```
# b2_steamdeck/config/steamdeck.yaml
steamdeck_audio: False
```

Verify connection:

```
ros2 topic echo /b2_unit_001/steamdeck_joy_teleop/cmd_vel
```

12 | Installation

This section covers the installation procedures for the B2W system.

> **Note:** This should already be configured by the MYBOTSHOP team.

12.1 | B2 Nvidia (Robot)

1. Create workspace:

```
sudo hostnamectl set-hostname b2-unit-366
sudo mkdir /opt/mybotshop
sudo chown -R unitree:unitree /opt/mybotshop
```

2. Set timezone:

```
sudo timedatectl set-timezone Europe/Berlin
sudo date -s "$(wget --method=HEAD -qSO- \
  --max-redirect=0 google.com 2>&1 | \
  sed -n 's/^ *Date: *//p')"
```

3. Run installer:

```
cd /opt/mybotshop/src/mybotshop
./b2_install.bash
```

4. Build workspace:

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

5. Add to .bashrc:

```
source /opt/ros/humble/setup.bash
source /opt/mybotshop/src/mybotshop/b2_bringup/\
config/setup.bash
```

6. Install services:

```
ros2 run b2_bringup startup_installer.py
```

7. Configure secondary IP for camera stream:

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

Add the following configuration:

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eth0:
      dhcp4: false
      addresses: [192.168.123.164/24,192.168.123.170/24]
      nameservers:
        addresses: [114.114.114.114,8.8.8.8]
```

Apply the configuration:

```
sudo netplan apply
```

12.2 | Host PC

1. Source ROS and build:

```
source /opt/ros/humble/setup.bash
colcon build --symlink-install
```

2. Configure network:

   Connect to 192.168.123.x network with static IP:

   - Address: **192.168.123.51**
   - Netmask: **24**

3. Source setup:

```
source /opt/mybotshop/src/mybotshop/b2_bringup/\
config/setup.bash
```

4. Verify connection:

```
ping 192.168.123.164
ssh -X unitree@192.168.123.164
# Password: Unitree0408
```

12.3 | Sync Host Computer to Robot

To sync your development workspace to the robot:

```
rsync -avP -t --delete -e ssh src \
  unitree@192.168.123.164:/opt/mybotshop
```

13 | Debugging

This section covers debugging tools and techniques for the B2W ROS2 system.

13.1 | Check Transforms (TF Tree)

To debug TF tree:

```
ros2 run rqt_gui rqt_gui --ros-args \
  --remap tf:=/b2_unit_001/tf \
  tf_static:=/b2_unit_001/tf_static
```

13.2 | Visualize URDF

To view the robot's URDF model:

```
ros2 launch b2_description b2w_description.launch.py
```

13.3 | Start-up Job

The B2W uses robot_upstart services for automatic launch of ROS2 nodes.

13.3.1 | Check B2 Services

```
# List all b2 services
systemctl list-units --type=service | grep b2

# Check specific service status
sudo systemctl status b2-webserver
sudo systemctl status b2-hardware
sudo systemctl status b2-livox-mid360
```

13.3.2 | Service Status Indicators

- **Red marker** - The startup job has failed
- **Green marker** - Everything is working correctly

- **Grey marker** - The service has not started yet

13.3.3 | Restart Services

```
# Restart B2 services
sudo systemctl restart b2-hardware
sudo systemctl restart b2-webserver
```

Warning: Do not run duplicate nodes! If a service is running (e.g., `b2-hardware` is active), the corresponding launch file is already running. Manually launching the same file will create duplicate nodes, causing conflicts.

13.3.4 | Check Before Manual Launch

Always run this before manually launching any ROS2 nodes:

```
# Quick check - shows all active b2 services
systemctl list-units --type=service --state=active | grep b2

# Or check specific service
systemctl is-active b2-hardware && \
  echo "SERVICE RUNNING - DO NOT LAUNCH MANUALLY"
```

13.4 | Robot Upstart Services

The B2W uses the `robot_upstart` package to create systemd services that start automatically on boot, restart on failure, and log to the system journal.

13.4.1 | Available Services

Service Name	Launch File	Auto-Start
b2-hardware	hardware.launch.py	Yes
b2-statepublisher	state_publisher.launch.py	Yes
b2-domain-bridge	bridge.launch.py	Yes
b2-front-video	front_video.launch.py	Yes
b2-rear-video	rear_video.launch.py	Yes
b2-twistmux	twistmux.launch.py	Yes
b2-webserver	webserver.launch.py	Yes
b2w-description	b2w_description.launch.py	Yes
b2-livox-mid360	livox_mid360.launch.py	Yes
b2-pan-tilt	dynamixel.launch.py	Yes

13.4.2 | Service Management Commands

```
# Check service status
sudo systemctl status b2-<service_name>

# Start/stop/restart service
sudo systemctl start b2-<service_name>
sudo systemctl stop b2-<service_name>
sudo systemctl restart b2-<service_name>

# Enable/disable service at boot
sudo systemctl enable b2-<service_name>
sudo systemctl disable b2-<service_name>

# Reload systemd after config changes
sudo systemctl daemon-reload
```

13.4.3 | View Service Logs

```
# View recent logs for a service
journalctl -u b2-hardware -n 50

# Follow logs in real-time
journalctl -u b2-webserver -f
```

```
# View logs since last boot
journalctl -u b2-hardware -b
```

13.4.4 | Troubleshooting Duplicate Node Issues

If you accidentally ran duplicate nodes:

```
# Kill all ROS2 nodes in namespace
pkill -f "ros2.*b2_unit_001"

# Restart the service
sudo systemctl restart b2-hardware
```

13.5 | Installing Services

Run the startup installer to configure all systemd services:

```
ros2 run b2_bringup startup_installer.py
```

This installs services using `robot_upstart` with:

- ROS Domain ID: 10
- RMW Implementation: CycloneDDS
- Workspace setup: `/opt/mybotshop/src/mybotshop/b2_bringup/config/setup.bash`

13.6 | Troubleshooting Common Issues

13.6.1 | No topics visible

```
# Verify ROS_DOMAIN_ID
echo $ROS_DOMAIN_ID  # Should be 10

# Check network
ping 192.168.123.165
```

13.6.2 | Dynamixel communication error

```
# Check USB device
ls -la /dev/b2/dynamixel

# Reboot servos
ros2 service call /dynamixel/reboot std_srvs/srv/Trigger
```

13.6.3 | LiDAR not connecting

```
# Verify network
ping 192.168.123.120  # Front
ping 192.168.123.121  # Rear
```

13.6.4 | Service Environment Issues

```
# Check what environment the service sees
sudo systemctl show b2-hardware \
  --property=Environment

# Verify startup.bash sources correctly
cat /opt/mybotshop/src/mybotshop/b2_bringup/config/setup.bash
```

14 | Autonomous Mobile Robot Safety Guidelines

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

14.1 | Work Area Safety

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

14.2 | Electrical Safety

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

14.3 | Navigation Safety

- Implement obstacle detection and avoidance systems to prevent collisions with objects, people, or other robots.
- Define and enforce safety zones within the robot's operational area to minimize the risk of unintended interactions with personnel or other equipment.
- Regularly calibrate and test the robot's navigation sensors and systems to ensure accurate and reliable performance.

14.4 | Emergency Response

- Install an emergency stop mechanism to quickly halt the robot's operation in case of unforeseen circumstances or emergencies.
- Clearly mark and communicate emergency stop locations throughout the robot's operational area.

- Conduct regular emergency response drills to ensure personnel are familiar with procedures for handling unexpected situations.

14.5 | Data Security and Privacy

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

14.6 | Human Interaction Safety

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

14.7 | Residual Risks

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

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

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

15 | Robotic Manipulator Safety Guidelines

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:

15.1 | 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.

15.2 | 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.

15.3 | Manipulation Safety

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

15.4 | Emergency Response

- Install an emergency stop mechanism to swiftly halt manipulator operation in unforeseen circumstances or emergencies.
- Clearly mark and communicate emergency stop locations within the manipulator's operational area.

- Conduct regular emergency response drills to ensure personnel are familiar with procedures for handling unexpected situations during manipulator operation.

15.5 | Data Security and Privacy

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

15.6 | Human Interaction Safety

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

15.7 | Residual Risks

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

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

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

Resources and Contact Information

For your convenience, we have compiled the most important resources related to the robot platform. Each link can be accessed directly or by scanning the corresponding QR code.

Documentation

Our official documentation contains detailed setup instructions, usage guidelines, and troubleshooting information. Access the latest version here: https://www.docs.quadruped.de/index.html

Forum

Connect with other developers and researchers in our community forum. Here you can ask questions, share experiences, and stay up to date: https://forum.mybotshop.de/

Rentals

If you would like to test or temporarily use a robot, we also offer rental services. Browse our available systems here: https://www.mybotshop.de/Vermietung_3

Newsletter

Stay informed about product updates, events, and special offers by subscribing to our newsletter: https://www.mybotshop.de/Newsletter_1

Contact Information

- Vertrieb / Sales: +49 (0) 2271 588 93 10 — `sales@mybotshop.de`
- Support / After Sales: +49 (0) 2271 588 93 20 — `support@mybotshop.de`
- Buchhaltung / Accounting: +49 (0) 2271 588 93 80 — `accounting@mybotshop.de`
- Allgemeine Anfragen / General inquiries: +49 (0) 2271 588 93 0 — `info@mybotshop.de`

MYBOTSHOP GMBH

RMA SUPPORT

How to Raise a Return Merchandise Authorization (RMA) Case

At QUADRUPED Robotics, we are committed to providing excellent support to resolve your issues 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 an RMA number along with a return address or a shipping label.

Important Shipping Information

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 an RMA number. Parcels sent without an RMA number will be declined and returned to the sender.
- Do **not ship goods** to QUADRUPED Robotics GmbH's Office in Leverkusen!

Help Resources

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

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