Robotnik

Appendix:
RB-Kairos+ with Franka Emika Arm

Revision v.1.1

Contents

1. Introduction

This document is applicable to all RB-Kairos+ with a Franka Emika arm and its purpose is to provide information about the mounting, initializing and control of the Franka arm installed on the RB-Kairos+ robot.

Depending on the machine purpose and components, the RB-Kairos+ can be adapted to different robotic arms:

- Franka Production 3 arm: This arm is used with robots with PLC and safety lasers. RB-Kairos+ manuals are applicable, taking into account this appendix and the modifications referring to the robotic arm.

- Franka Research 3 arm: This arm can be installed in Research RB-Kairos+, without safety PLC. Please take into account the specific documentation of this robot, and pay special attention to possible risk situations. The end user must perform a risk assessment of the end application.

The RB-Kairos+ user manual and maintenance manual, the control interface manual and the developer manual are applicable to RB-Kairos+ with Franka Emika arm, taking into account the considerations listed in this document.

Please read the safety warnings included in the user manual and Research RB-Kairos+ documentation if applicable before interacting with the robot.

For further information about the arms, please consult the Franka Emika official manuals.

2. Arm Mounting

Once the robot is turned off and out of the box, you can proceed to mount the robotic arm on the RB-Kairos+ base. Please follow the instructions detailed below:

1.Unscrew the arm fixing screws
Unscrew the 4 arm fixing screws from the rear top cover (the one that is closer to the rear panel).

2. Place the arm on the top cover
Approach the arm to the mounting position, align the arm base with screw holes, and place it on the top cover.

3.Screw the arm to the top cover
Tighten the four screws to 10Nm torque.

4. Connect the robot cables
Identify the arm and tool cables on the RB-Kairos+ (the ones that pass through the cable entry frame) and connect them to the robotic arm and the tool.

3. Arm Power Selector

The robot has an added selector on the rear panel that allows to power on and off the robot.

> **WARNING**
>
> The arm must be switched on only after robot base initialization. Otherwise the robot can experience a power consumption peak too high for the system, and will not initialize properly.

4. Control

RB-Kairos+ and its software applications can be controlled and monitored through different systems:

- RB-Kairos+ is equipped with the Robotnik Human Machine Interface (HMI), an internal web application running locally on the robot. For further information and instructions about the HMI, please consult the RB-Kairos+ HMI manual.
- The robotic arm can be controlled through its Franka Desk HMI. For further information, please refer to Franka Emika official documentation.
- For advanced developers, RB-Kairos+ can be controlled through ROS programming. For further information, please refer to the RB-Kairos+ ROS developer manual.

In the following chapters it is described how to connect to the robot and software credentials, regardless of the software used.

4.1. Users and passwords

Franka arm admin	
User	root[1]
Password	`R0b0tn1Kkairos`

Franka arm safety	
User	`safety`
Password	`R0b0tn1Kkairos`

1. For older versions the user is *robot*

4.2. Network

Device	IP address
Franka Arm	192.168.0.210

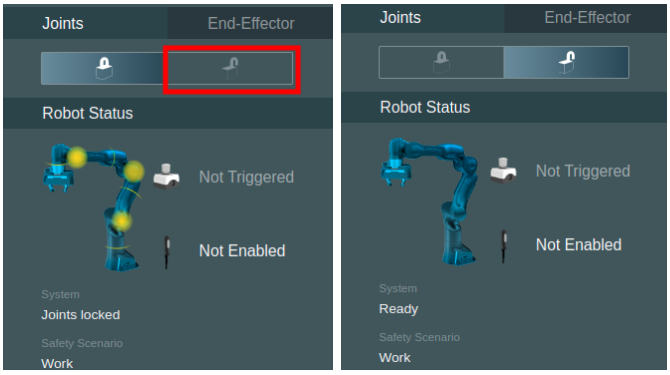
4.3. Arm Booting

The Franka arm uses its own interface in order to send and receive commands. Before controlling the arm using ROS, some steps are required:

1. Switch on the arm: Switch the arm power selector (inverter ON/OFF) in the robot rear panel to ON.

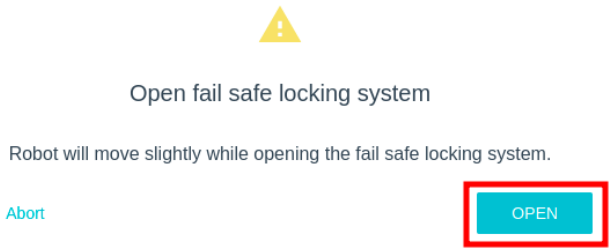
2. Connect to the network of the robot. Open a web browser and introduce the address of the Franka HMI: The IP of the arm is **192.168.0.210.** Introduce the franka arm admin credentials, listed in the Users and passwords section.

3. Wait until the Desk interface loads. Then, release the arm brakes by clicking on the open padlock.



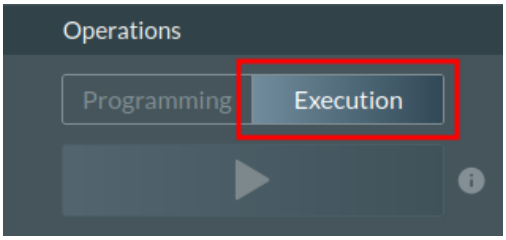
Franka release brakes

A safety message will appear asking for validation, to double check the action.

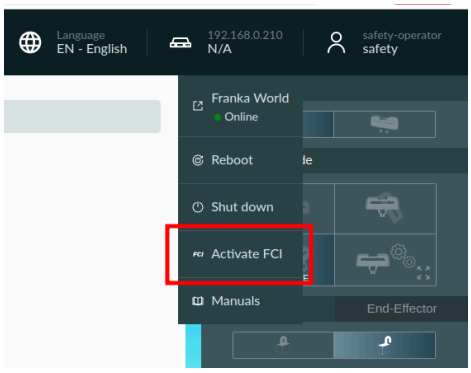


Franka release brakes pop-up

4. Set Execution mode in the Operations panel. Leds will be blue.

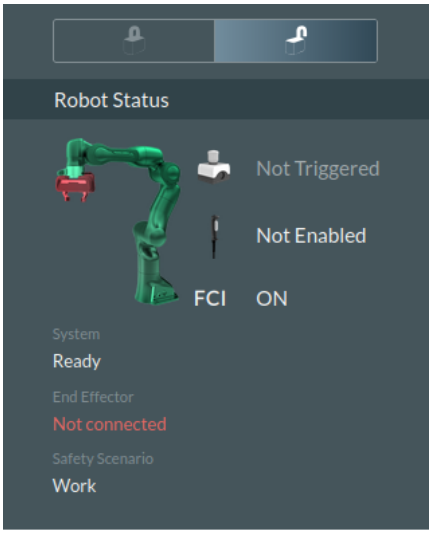


5. Click on Activate FCI in order to control the arm from ROS.



FCI feature

In case of having a Franka Hand **the end effector will switch to Not connected.** This is normal behavior.



6. Make sure that the robot base is not in emergency state. If the emergency button is pressed the arm will not move.

7. Connect to the robot base via ssh and relaunch the bringup script. This step is needed in order to allow the real time kernel to boot properly on the systemd services.

```
ssh robot@192.168.0.200
./bringup.sh
```

8. Make sure that the manipulation screen exists and the nodes are running. In case of error, launch the bringup script.

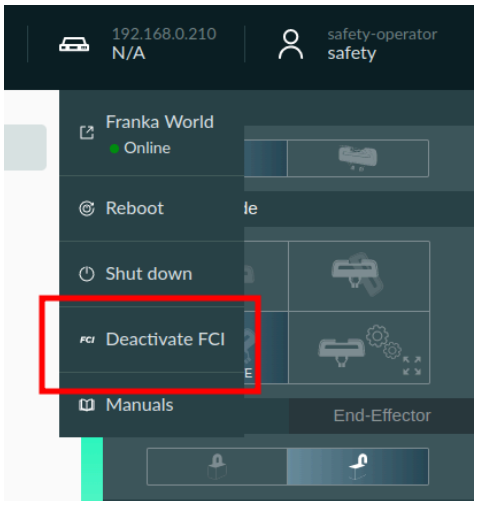
```
screen -r manipulation
```

9. Now the arm is ready to be controlled from ROS.

4.4. Arm shutting down

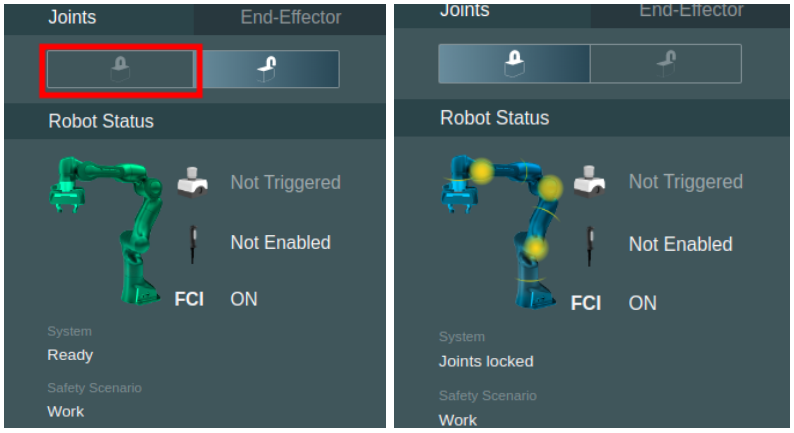
In order to turn off the arm safety follow these steps:

1. Go to the Franka Desk. Set the IP 192.168.0.210 into the address bar of a web browser inside of the network of the robot.
2. Deactivate the FCI mode if it was enabled. Click on Deactivate FCI:



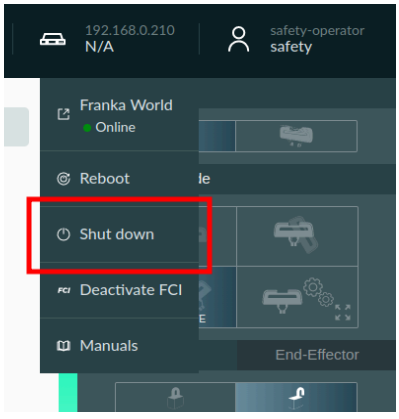
Deactivate FCI

3. Lock the joints by pressing on the close padlock:



Franka brakes

4. Click on the Shut down option:



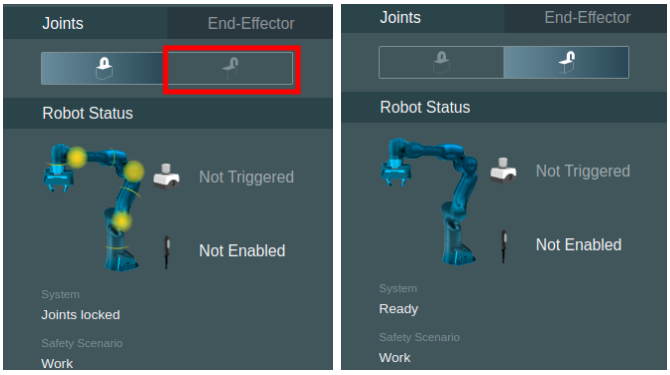
Arm shut down

5. Wait a few minutes and power off the arm using the arm power selector.

4.5. Arm free drive

The arm can be moved in free drive mode by a user. These are the steps:

1. Release the arm brakes by clicking on the open padlock.

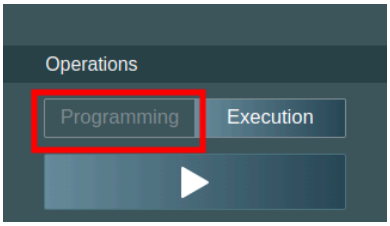


Franka release brakes

2. Set Guiding Mode as FREE MODE for better control.

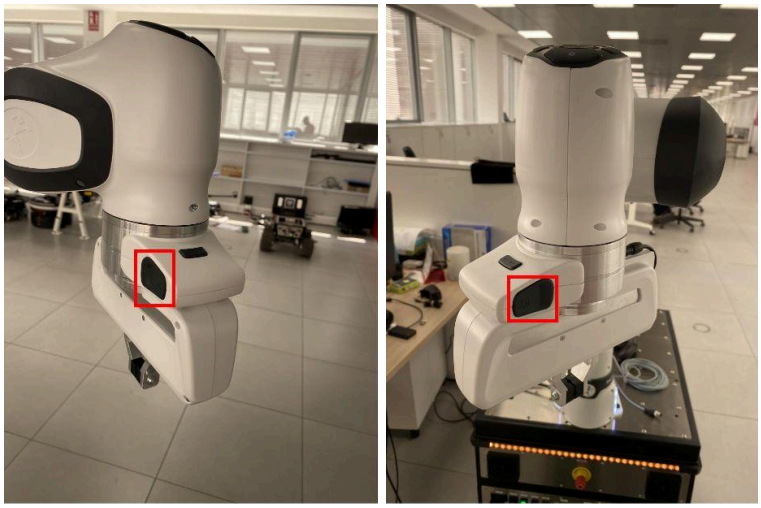
Guiding Mode panel

3. Set Programming mode in the Operations panel. Leds will be white.



Operations panel

4. Move the arm by **pressing at the same time the two buttons** that are in the end effector of the arm.



Free drive buttons

4.6. ROS Control

4.6.1. Available controllers

Once the arm has been booted, it is ready to receive commands from ROS. The arm can be used with a position or an effort controller, but they can not work at the same time.

- **position_joint_trajectory_controller**: quick movements but sometimes the joints oscillate for the control calibration.
- **effort_joint_trajectory_controller**: smooth movements but it takes more time to reach a point.

By default, the arm works with the effort_joint_trajectory_controller. To change the controller open the franka_complete.launch file:

```
gedit
~/catkin_ws/src/robot_packages/robot_bringup/launch/manipulation/arm/franka/franka_complete.launch
```

Replace the controller by **position_joint_trajectory_controller:**

```
23
24    <rosparam command="load" file="$(find robot_bringup)/config/manipulation/arm/franka/franka_default_controllers.yaml" subst_value="true" />
25    <node name="state_controller_spawner" pkg="controller_manager" type="spawner" respawn="false" output="screen" args="franka_state_controller effort_joint_trajectory_controller"/>
26    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" output="screen"/>
27    <node name="joint_state_publisher" type="joint_state_publisher" pkg="joint_state_publisher">
28      <rosparam if="$(arg load_gripper)" param="source_list">[franka_state_controller/joint_states, franka_gripper/joint_states] </rosparam>
29      <rosparam unless="$(arg load_gripper)" param="source_list">[franka_state_controller/joint_states] </rosparam>
30      <param name="rate" value="30"/>
31    </node>
32
33  </group>
34
```

franka_complete.launch file

Launch again the software to apply the changes:

```
./bringup.sh
```

4.6.2. Control status

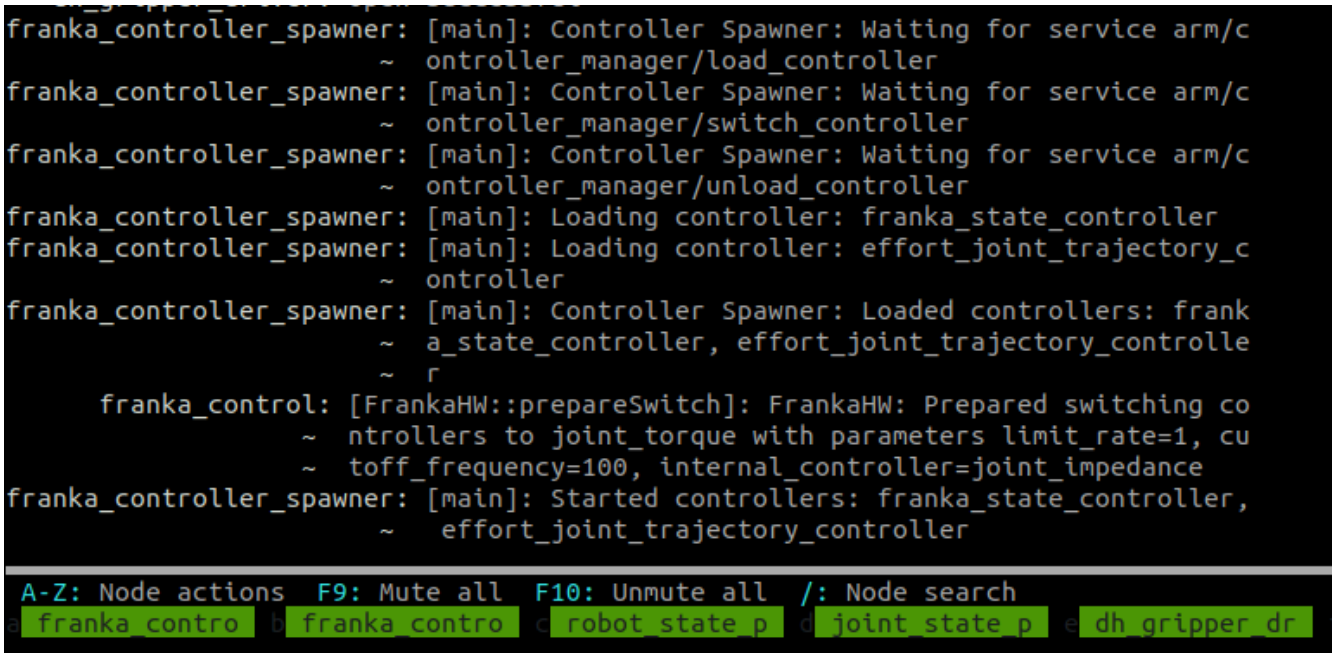
In order to check that the ROS Franka controller is working properly, access the robot base and check the logs.

1. Connect to the robot base via ssh.

```
ssh robot@192.168.0.200
```

2. Check that there are no errors in the log of the Franka arm in ROS. You can exit from the screen by holding Ctrl+A and pressing D.

```
screen -r manipulation
```

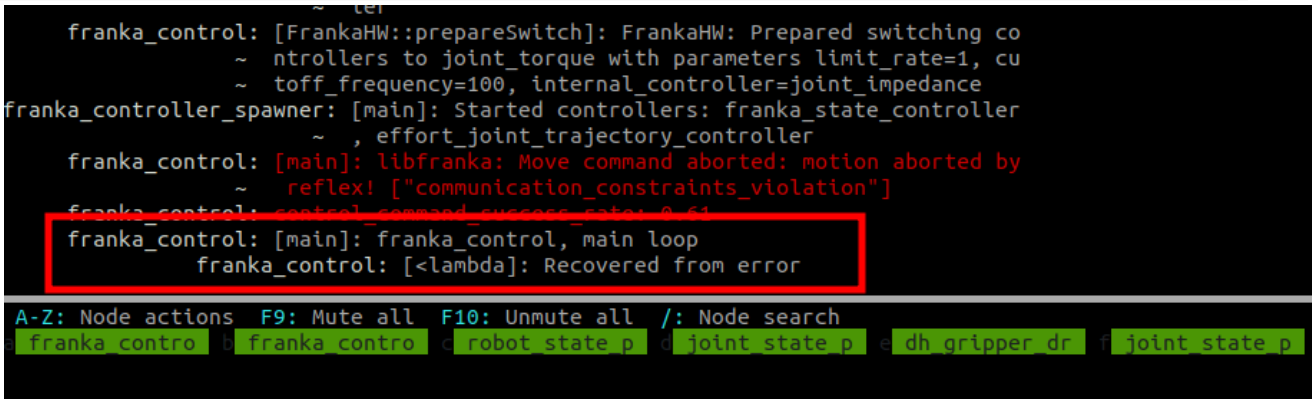


Franka manipulation screen without errors

4.6.2.1. Error recovery

If an error appears, for example an invalid joint position sent by ROS, the arm controller will not receive any more commands until the arm is recovered by calling the error_recovery action. Press TAB twice to complete the message.

```
rostopic pub -r 1 /robot/arm/franka_control/error_recovery/goal
franka_msgs/ErrorRecoveryActionGoal + TAB + TAB
```



Error recovery

4.6.2.2. Common error messages

These are the most common errors that can appear in the manipulation screen.

4.6.2.2.1. Move command rejected

```
libfranka: Move command rejected: command not possible in the current mode!
```

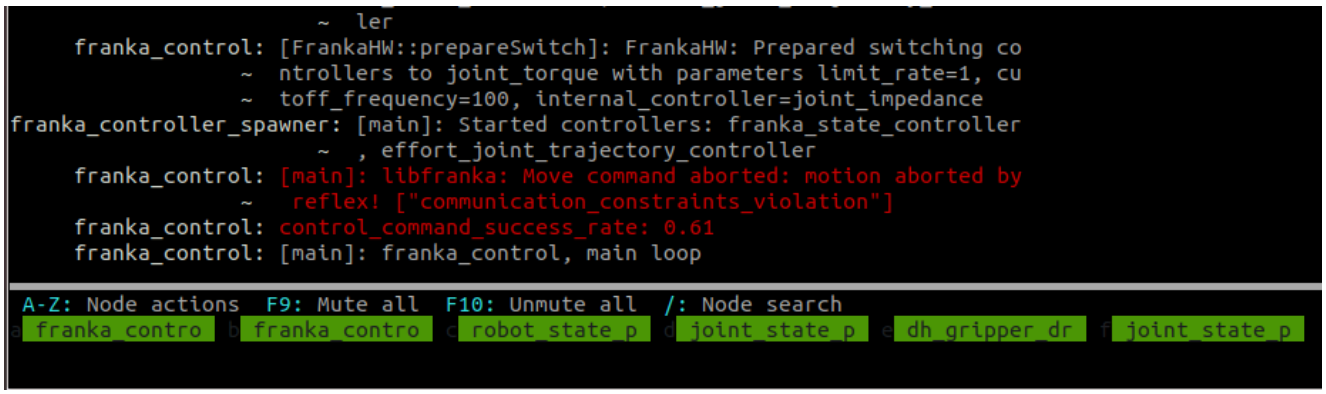
The robot base or the arm is in emergency mode. When that happens the PLC of the robot base sends a signal to the arm in order to disable the control. The arm will not move. Check the following points:

- Check if the wireless emergency button is connected. **Each time that the robot base turns on, the wireless button needs to be pressed and released at least one time.**
- If the robot base is working with the security enabled, check that the lasers do not detect nearby obstacles.
- Check if the emergency button of the robot base is released.
- Check if the blue button of the robot is turned off. This button must be pressed each time that the robot is released from an emergency.

4.6.2.2.2. Move command aborted

```
[main]: libfranka: Move command aborted!
```

```
[main]: libfranka: Move command aborted: motion aborted by reflex!
["communication_constraints_violation"]
```



Move command aborted error

These errors appear when the joint movements can not be done from ROS. It usually happens with Moveit due to the acceleration or joint limits.

4.6.2.2.3. Hardware error

Finally, if the arm joints are moving quickly, the arm will stop for security reasons. In that case, the arm led will be red and on the Franka Desk will appear the following error:

```
Franka Desk: hardware error
```

Release the brakes of the arm and activate the FCI again. Secondly, relaunch the manipulation screen. Press Ctrl + C in order to shutdown it, then press R to launch it again.

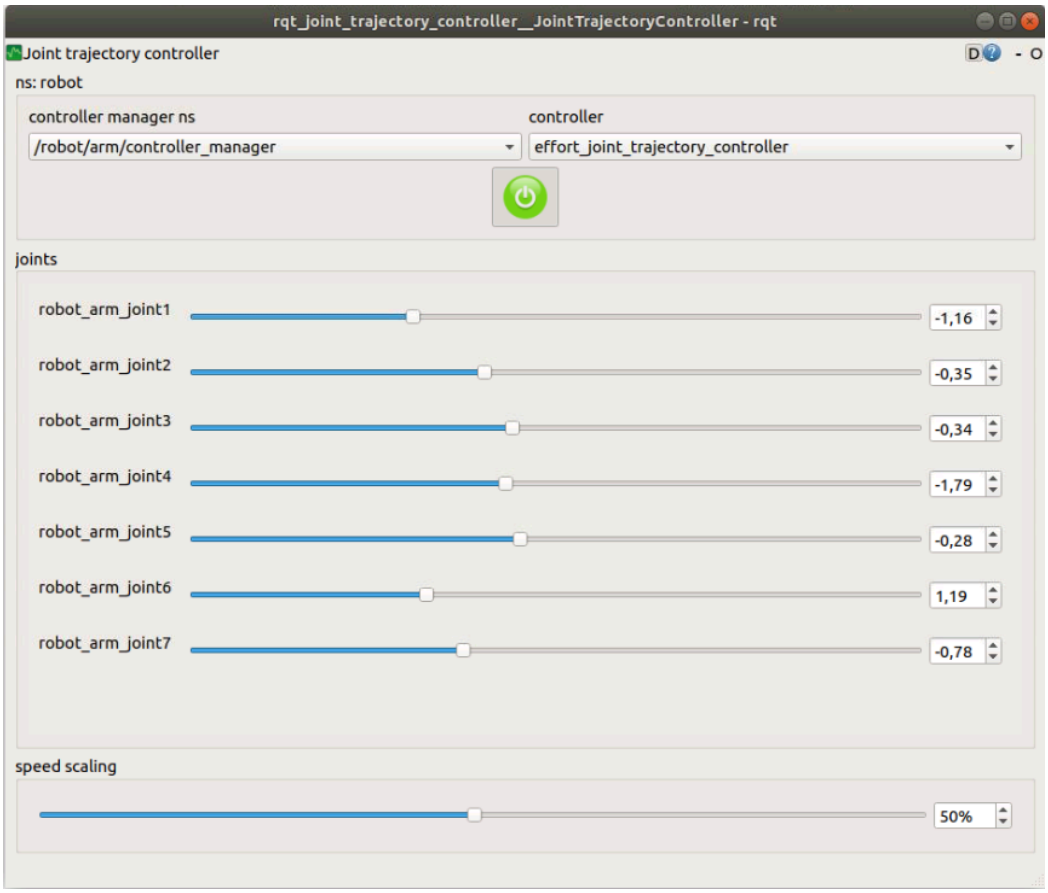
4.6.3. Joint by joint control

Make sure that the arm is ready and the manipulation screen is working without errors. Inside of the robot base via Remmina or Teamviewer, open a terminal. Then open the rqt plugin:

```
ROS_NAMESPACE=robot rosrun rqt_joint_trajectory_controller
rqt_joint_trajectory_controller
```

Select the namespace and the controller of the arm. Then click on the red button to control the arm joint by joint.

- **controller_manager_ns:** /robot/arm/controller_manager
- **controller:** effort_joint_trajectory_controller or position_joint_trajectory_controller



Joint by joint control

4.6.4. Moveit control

The control of the Moveit can be launched using the following command. The **name** of the package **depends on the arm** model.

```
ROS_NAMESPACE=robot roslaunch rbkairos_franka_moveit demo.launch
```

By default, Moveit works with the effort_joint_trajectory_controller. To change the controller open the ros_controllers.aml file:

```
gedit /home/robot/catkin_ws/src/rbkairos_franka_moveit/config/ros_controllers.yaml
```

Replace the controller by **arm/position_joint_trajectory_controller:**

```yaml
controller_list:
  - name: arm/effort_joint_trajectory_controller
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    default: True
    joints:
      - fr3_joint1
      - fr3_joint2
      - fr3_joint3
      - fr3_joint4
      - fr3_joint5
      - fr3_joint6
      - fr3_joint7
```

ros_controller.yaml file

4.7. Franka Hand

Franka Hand is a fully-integrated electrical two-finger parallel gripper produced by Franka Emika. No external cabling is needed. Communication and power supply for the Hand is established via the flange connector of the Arm.

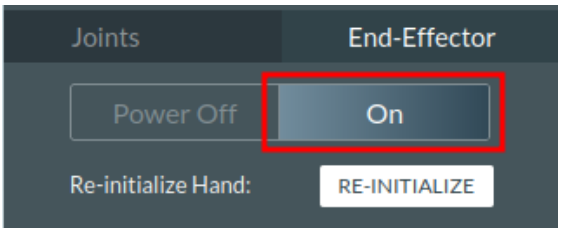
Franka Hand

4.7.1. Hand booting

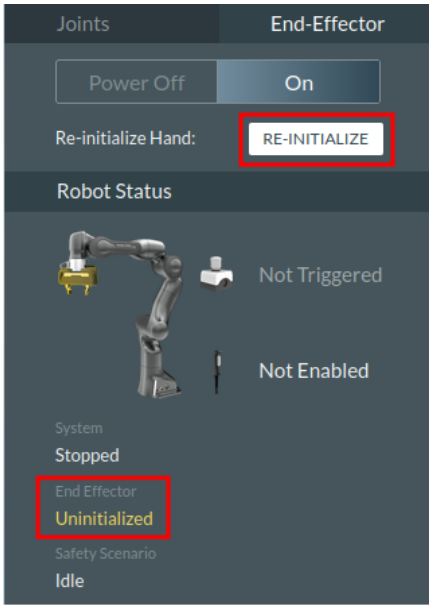
By default the Franka Hand boots automatically. In case to boot manually follow the following steps:

1. Make sure that the arm is booted following the steps of the Arm booting section.

2. Turn on the End-Effector:



3. Re-initialize the hand:



4.7.2. Hand free drive

The Franka hand can be moved in free drive mode by a user. These are the steps:

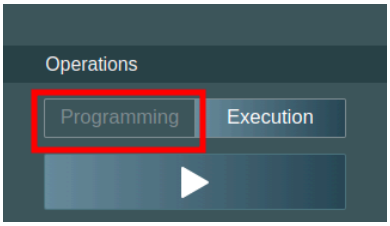
1. Make sure that the arm is in free drive mode following the steps of the Arm botting section.

2. Click on the gripper icon in the Pilot Mode panel

5. Set Guiding Mode as FREE MODE for better control.

Guiding Mode panel

6. Set Programming mode in the Operations panel. Leds will be white.



Operations panel

7. Move the gripper with your hands or use the following arm selector to open and close the gripper.

Open/Close Buttons

4.7.3. ROS control

You can use different ROS actions to control the gripper. Here we show the available actions and their objectives.

Note: make sure that the FCI interface is running before controlling the hand using ROS. **Follow the Arm booting** section to enable it.

The width range goes from 0.0 (completely closed) to 0.08 m (completely open) and the speed ranges from 0.0 to 0.05 m/s.

The move action is able to open or close the gripper completely.

```
rostopic pub /robot/arm/franka_gripper/move/goal + TAB + TAB
```

```
rostopic pub --once /robot/arm/franka_gripper/move/goal \
        franka_gripper/MoveActionGoal \
        "goal: { width: 0.03, speed: 0.1}"
```

The stop action can cancel the action that the gripper is executing.

```
rostopic pub /robot/arm/franka_gripper/stop/goal + TAB + TAB
```

```
rostopic pub --once /robot/arm/franka_gripper/stop/goal \
        franka_gripper/StopActionGoal \
        "goal: {}"
```

Finally, the grasp action allows the gripper to grasp objects.

```
rostopic pub /robot/arm/franka_gripper/grasp/goal + TAB + TAB
```

```
rostopic pub --once /robot/arm/franka_gripper/grasp/goal \
        franka_gripper/GraspActionGoal \
        "goal: { width: 0.03, epsilon:{ inner: 0.005, outer: 0.005 }, speed: 0.1, force: 5.0}"
```

4.8. Troubleshooting

4.8.1. Arm topics not available

Make sure that the manipulation screen exists and the nodes are running.

```
screen -r manipulation
```

In case of error, press Ctrl+C to shutdown the nodes of the screen and press ‘r’ to launch them again.

Note: in the case of Franka Hand, disabling the FCI interface causes shutdown of the gripper and controller node. Every time that the FCI is disabled, the nodes of the manipulation screen have to be launched manually.

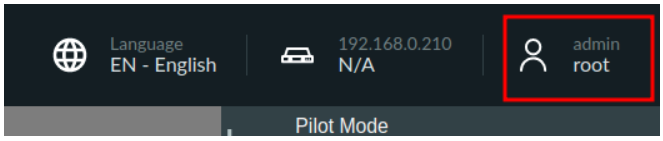
4.8.2. Joint position error detected

Sometimes the robot can lose its reference position because the FR3 and FP3 systems are equipped with extended safety functions. If this happens the following message will appear "Joint position error detected".

Joint position error

For solving this you will have to follow the next steps that appear in the Franka product manual. If you want more details about something, search for this manual.

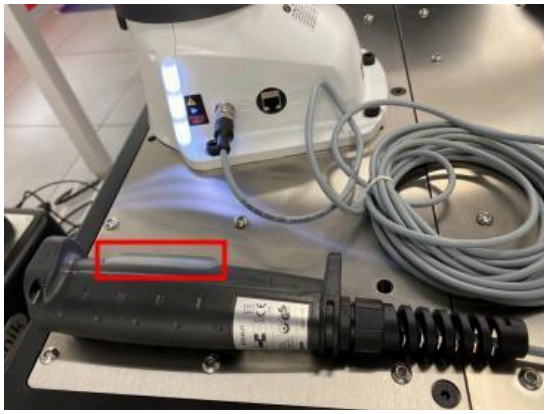
1. Change to the safety operator user. In order to do it, firstly you have to log out.



User change

2. Enter with the Franka safety user (check Users and passwords section).

3. Connect the external enabling device to the robot in the X4 port of the arm and press the "Start Recovery" button.

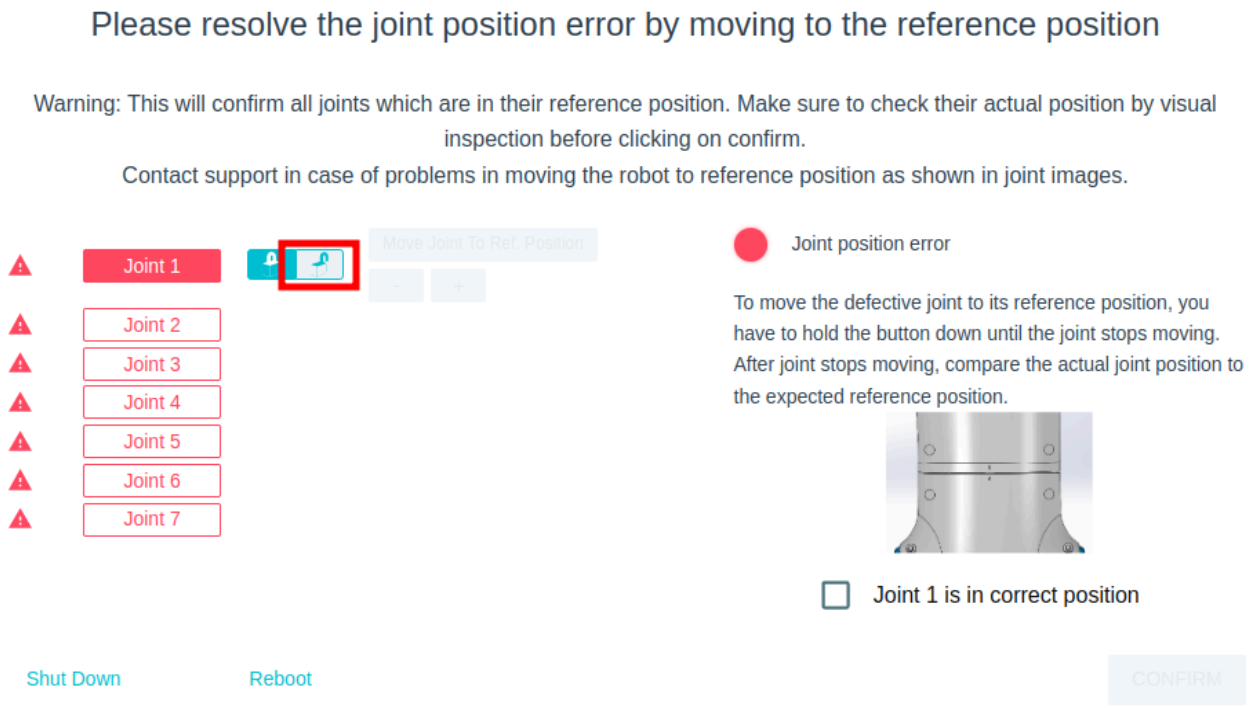


External enabling device

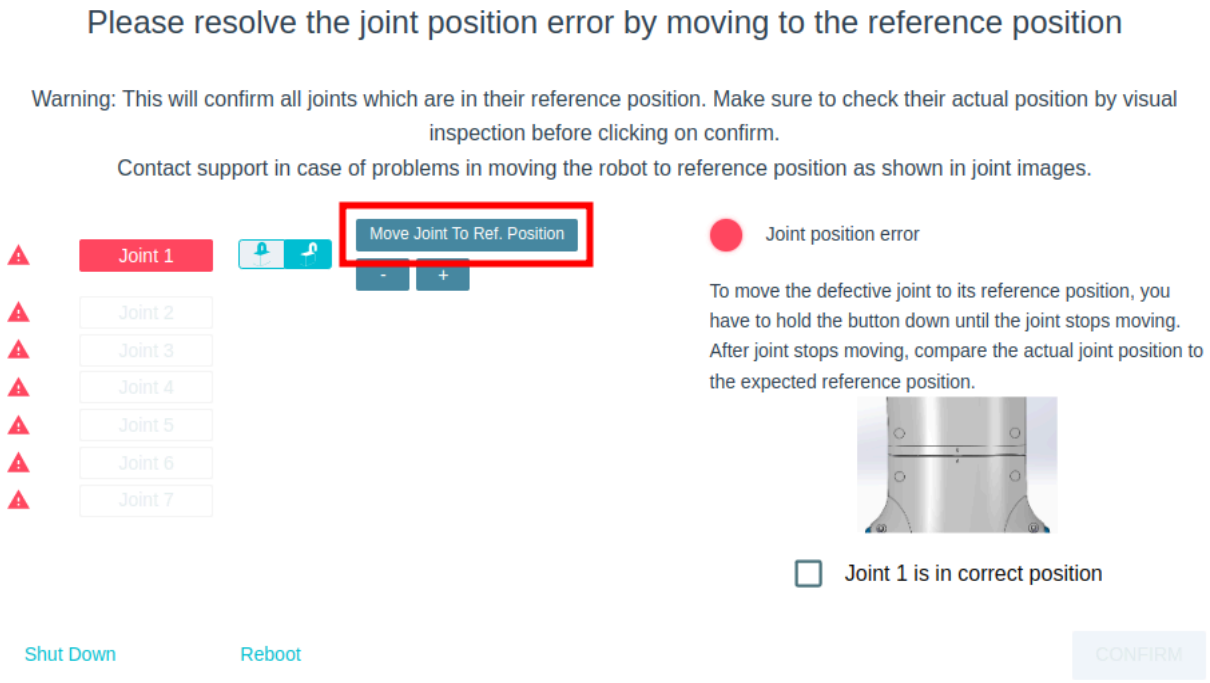
4. A position error will be displayed. Press “start recovery”.

Start recovery

5. Go joint by joint holding the enabling button, unlocking each joint and pressing the “Move Joint To Ref. Position” option until that joint goes to the reference position by itself.



Unlock joint

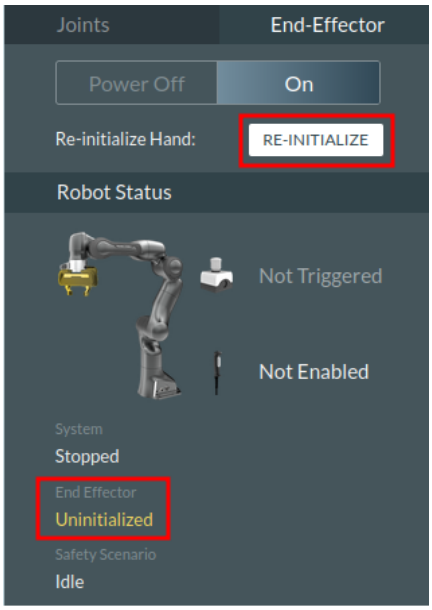


Move joint to reference position

6. Change another time to the admin user (check Users and Passwords step).

4.8.3. Franka Hand gripper not initialized

The griper does not initialize after a manual re-initialization. The FCI can be enabled but ROS commands are ignored.



Follow the steps of the Hand free drive section. Move the gripper manually, then reboot and re-initialize the gripper.

Research RB-Kairos+

This document is applicable to all RB-Kairos+ without safety lasers. This type of robot is designed for **research purposes only**, and must not be used for any other type of application.

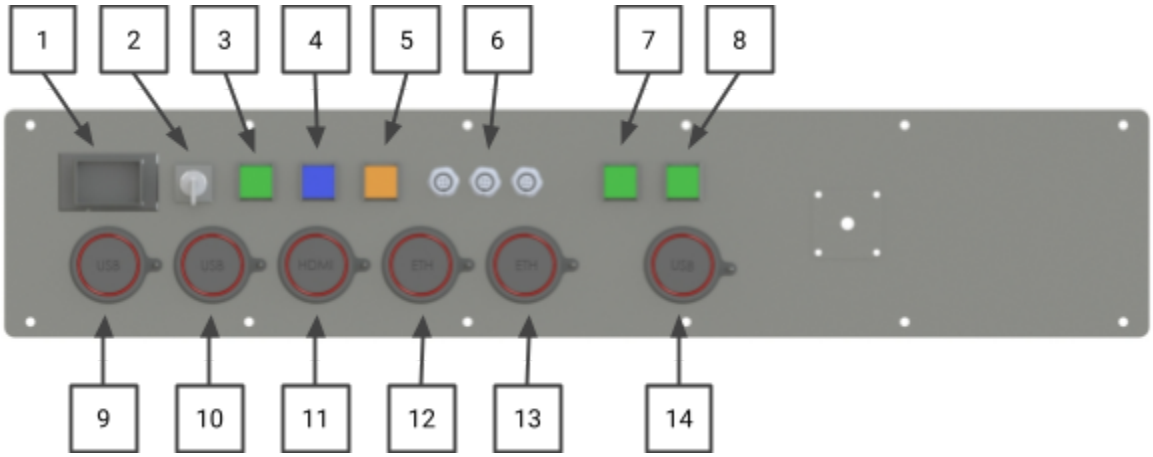
The end user or end integrator of the machine **must perform a risk assessment** of the end application, taking in account the added tool and the work environment, and take the appropriate actions.

	**WARNING** **Do not use the robot for non-research applications**. Please remember to perform a risk assessment of the end application of the robot, as end integrator of the machine. Adapt the working area and environment if necessary.
	The 2D lasers of the robot are not safety certified. Do not trust obstacle avoidance during robot navigation for safety purposes.
	The robot arm will move freely regardless of the presence of an obstacle or a person in the surrounding area of the robot. Please take the appropriate actions for avoiding damage to the environment or injure someone.

This robot has several differences with the standard RB-Kairos+. Nevertheless, the RB-Kairos+ user manual and maintenance manual, the control interface manual and the developer manual are applicable to RB-Kairos+ without safety lasers, taking into account the considerations listed in this document.

For further information, please contact Robotnik support.

1. Main components: rear panel



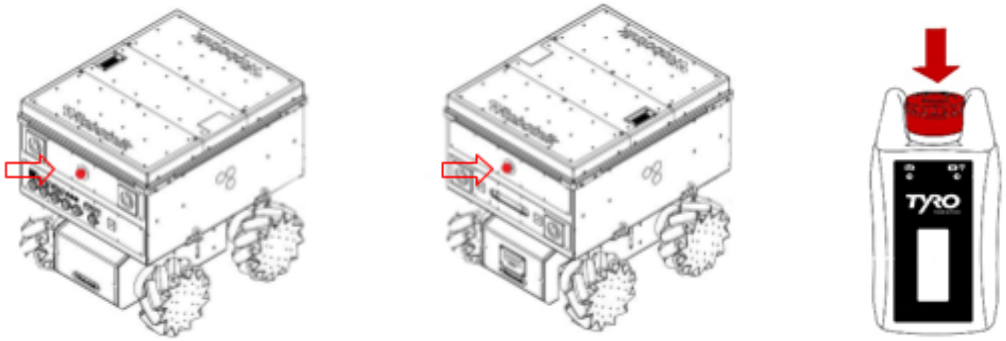
1. Manual charger connector
2. Main power selector (On/Off)
3. PC Robot button
4. Restart button
5. Brake Release button
6. Power connectors : 12V, 24V, VDC BAT
7. On UR arm button
8. Off UR arm button
9. USB 2.0 port to the base CPU
10. USB 2.0 port to the base CPU
11. HDMI port to the robot CPU
12. Wan Ethernet port
13. LAN Ethernet port
14. HDMI + USB port to the UR arm

Research RB-Kairos+ rear panel

2. Safety

In case of any emergency, the RB-Kairos+ has 2 available emergency stop buttons: one on the front and one on the back. Push any of them to stop the robot. Release it and rearm the robot to get back to normal operation.

Optionally, the robot can have a remote emergency stop device installed.



RB-Kairos+ emergency stops

3. Working modes

This robot does not have different working modes.

4. Brake release functionality

This robot has an added functionality for emergency situations: the brake release function.

In case of an e-stop button activation, the robot will perform an emergency stop procedure and will activate the wheel brakes, in order to avoid the movement of the robot. If it is not possible to recover the emergency stop in the current situation, and the robot has to be relocated for any reason, please follow the next steps:

1. During an emergency stop situation, press the *brake release* button placed on the rear panel.
2. With the button pressed, push the robot to the desired position.
3. Turn off the robot: after using the brake release function, a restart of the robot is needed for an appropriate behavior of the motor drivers.

5. Related standards and documentation

For this type of robots, Directive 2006/42/EC of European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC is not applicable.

Nevertheless, Robotnik robots are designed and built with a high quality standard compromise and accomplish as many european standards as possible according to the product specifications.

- **EN 60204-1:2006/A1:2019**. Safety of machinery - Electrical equipment of machines - Part 1: General requirements.
- **EN ISO 13850:2015**. Safety of machinery – Emergency stop function – Principles for design (ISO 13850:2015).
- **EN ISO 13849-1:2015**: Safety of machinery - Safety related parts of control systems - Part 1: General principles for design.
- **EN 1175-1:1998+A1:2010**. Safety of industrial trucks – Electrical requirements – Part 1: General requirements for battery powered trucks.
- **EN 60204-1:2006-6**. Safety of machinery – Electrical equipment of machines – Part 1: General requirements.

Control Interface

Manual

Revision v.1.2 - Noetic

Contents

1. Introduction

Welcome to the Control Interface manual. This document explains how to use the Robotnik Human Machine Interface (HMI) and the Robotnik Integration URCap (for UR arms) in order to interactuate with the robot in a simpler way .

This document is dedicated to the daily operator of the robot, with basic knowledge of informatics and programming.

Please, pay close attention to safety warnings of your robot before operating it.

For further information, you can consult the following documents of your robot:

- User manual of the robot
- Maintenance manual of the robot
- Developer manual
- Appendices

1.1. Contact

If you have any doubt or problem with your robot, please contact us by sending an email to support@robotnik.es, indicating the serial number of your robot. We are constantly improving and upgrading our products and services. Any feedback from users is welcome.

NOTE: You can get in contact with us either in English or Spanish.

1.2. Robot Types

This manual is applicable to all the Robotnik robots family under ROS development, except to the RB-Kairos+ 16e.

NOTE: This manual may include some images from different robots, as visual support of the applicable written information.

2. HMI

Robotnik Human Machine Interface (HMI) is a tool to monitor the robot status, control and perform localization and navigation actions in a simpler way. The HMI is an internal web application running locally on the robot.

2.1. Access

In order to access the HMI, you must be connected to the same network as the robot. There are two possible ways to do it:

- **Wifi**: most robots generate their own network wifi. The name of the network is the serial number of the robot. For example SHL00-XXXXXX_2_4G.
- **LAN**: connect an ethernet cable from the LAN port of the robot to your computer. The name of the network is the serial number of the robot. For example SHL00-XXXXXX_2_4G.

These are the standard credentials of the robot network:

Network name: SHL00-XXXXXX_2_4G (serial number of the robot, this is an example)
Password: R0b0tn1K

For further information or configuration variations please consult the specific information of your robot.

2.1.1. Windows users with Wifi

Click on the wireless icon, then the available networks will appear. The name of the network is the serial number of the robot.

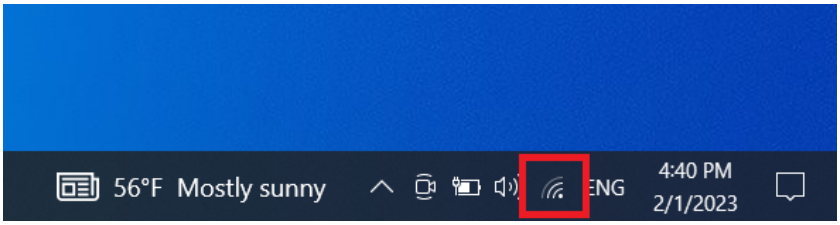


Figure 1. Windows wifi connection.

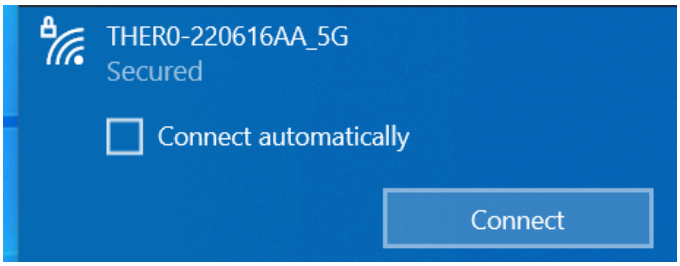


Figure 2. Windows wifi network.

2.1.2. Windows users with LAN

Click on the computer icon, then the available networks will appear. The name of the network is the serial number of the robot.

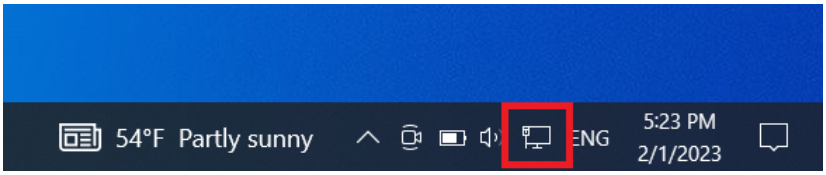


Figure 3. Windows ethernet connection.

Figure 4. Windows ethernet network.

⚠ It is recommended to disable the Wifi while the ethernet connection is setting.

Select the robot network to open the configuration panel.

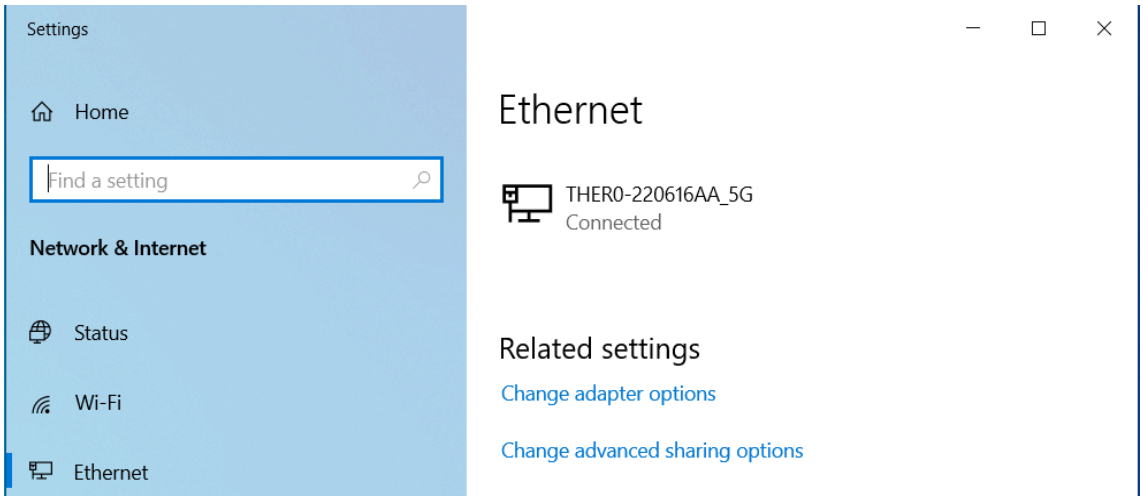


Figure 5. Windows ethernet configuration panel.

In the configuration panel, select the robot network to edit the IP of our laptop.

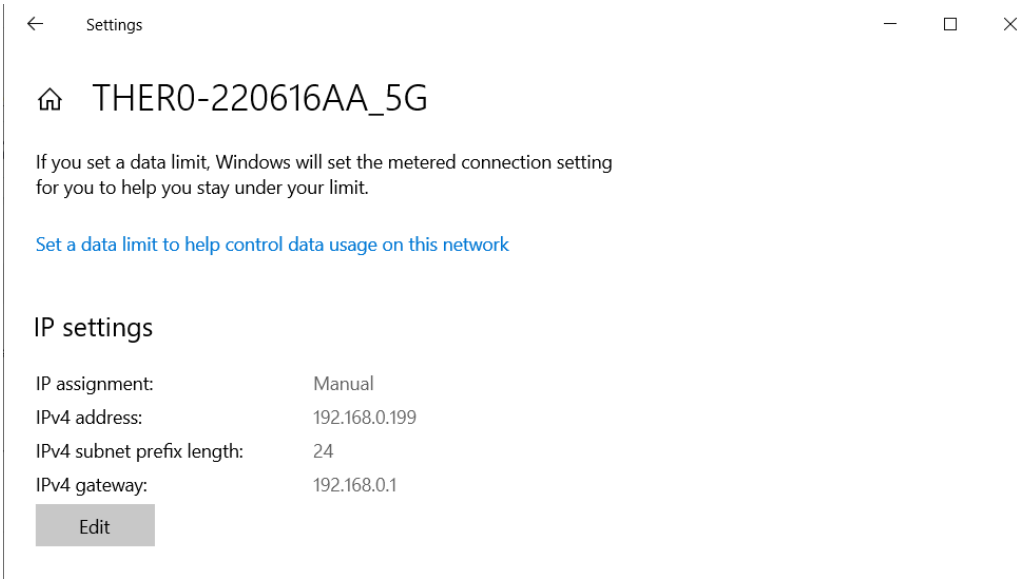


Figure 6. Windows ethernet configuration.

Set an IP for the laptop in the robot network. Make sure that the IP does not match any of the devices of the robot. A good IP candidate could be: 192.168.0.199.

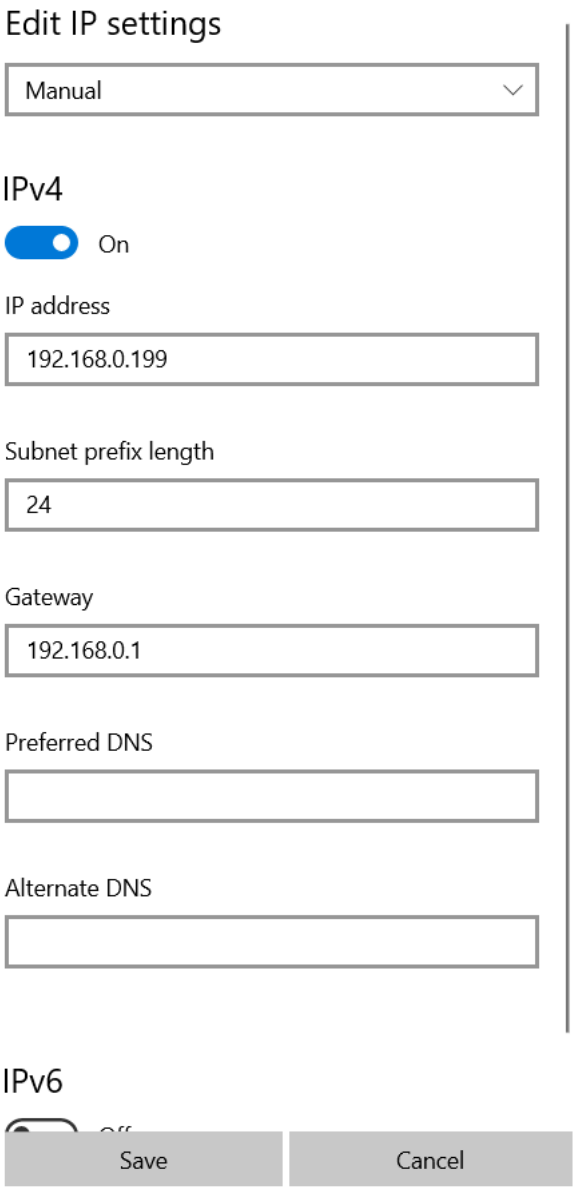


Figure 7. Windows ethernet settings.

Save the configuration and make sure the changes have taken effect. Otherwise, disconnect and connect the ethernet cable.

2.1.3. Linux users with Wifi

Click on the wireless icon, then select “Network”. The name of the network is the serial number of the robot.

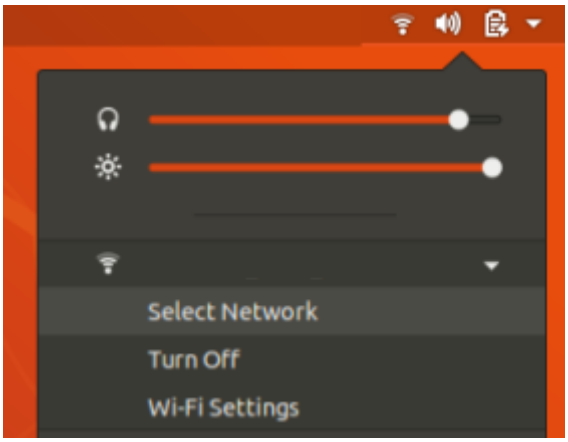


Figure 8. Linux wifi connection.

Figure 9. Linux wifi network.

2.1.4. Linux users with LAN

Click on the ethernet icon, then select “Wired Settings”. The name of the network is the serial number of the robot.

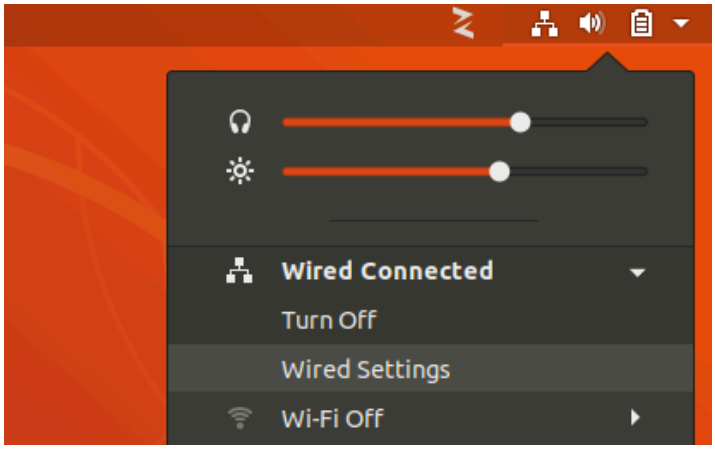


Figure 10. Linux ethernet connection.

⚠ In order to avoid connection problems, disable the Wifi meanwhile the ethernet connection is setting.

Select the robot network to open the Network Settings.

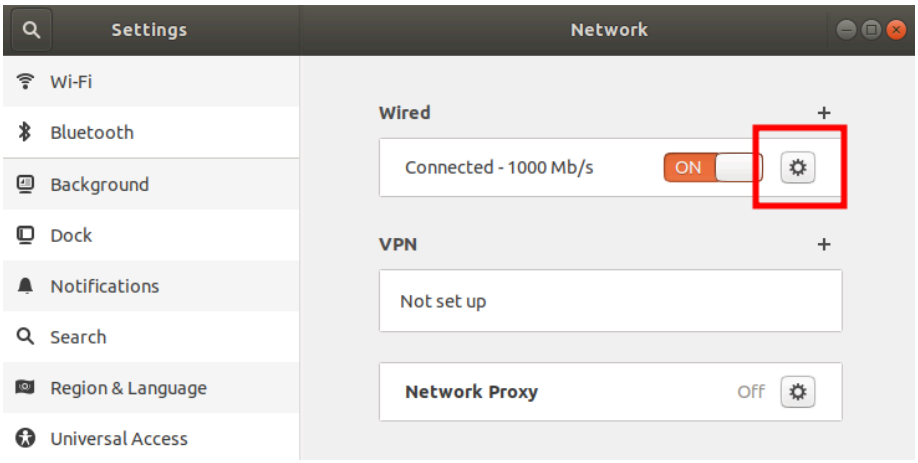


Figure 11. Linux ethernet network.

Click on the gear button in order to configure the IP of the laptop. Make sure that the IP does not match any of the devices of the robot. A good IP candidate could be: 192.168.0.199.

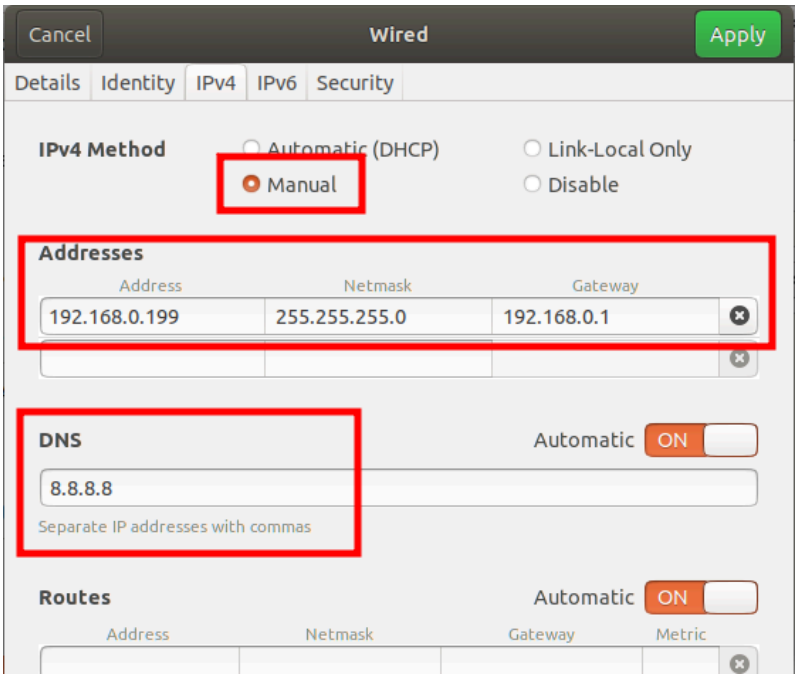


Figure 12. Linux ethernet settings.

Save the configuration and make sure the changes have taken effect. Otherwise, disconnect and connect the ethernet cable. This step can be done physically or from the Network Settings.

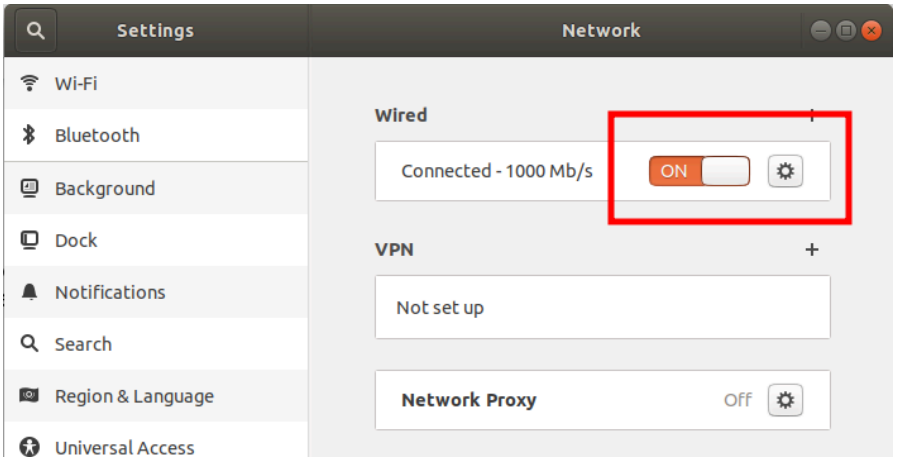


Figure 13. Linux ethernet reconnection.

In any case, make sure the changes have taken effect in the Details tab.

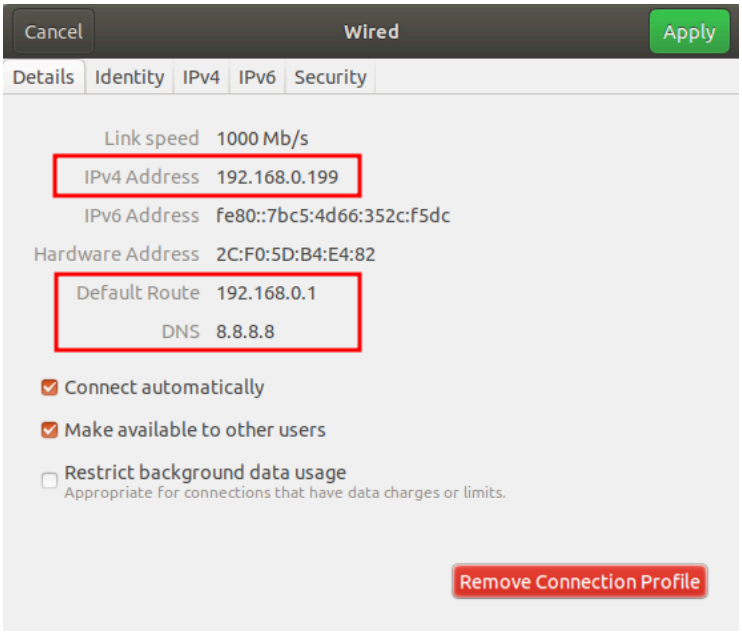


Figure 14. Linux ethernet final settings.

2.1.5. Webpage access

You can access the HMI through a web browser: is it recommended to use Firefox, although Google Chrome is also possible.

Figure 15. Available web browsers.

Once the connection to the robot network is established, you can access to the HMI website using the following URL:

http://192.168.0.200/robotnik_hmi

Open the web browser and place the URL in order to access to the HMI:

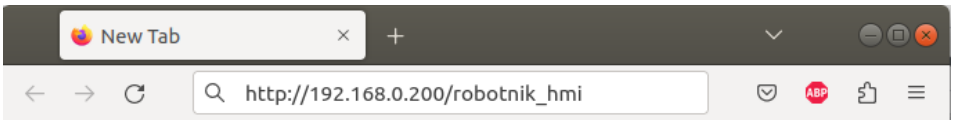


Figure 16. Web browser bar.

Note: if the robot is connected to another network, replace the robot computer IP with the IP assigned in that network: http://robot_ip/robotnik_hmi

2.2. Entry page

Once the laptop is connected to the robot network, the HMI will be accessible using the web browser. The first page will be the login page.

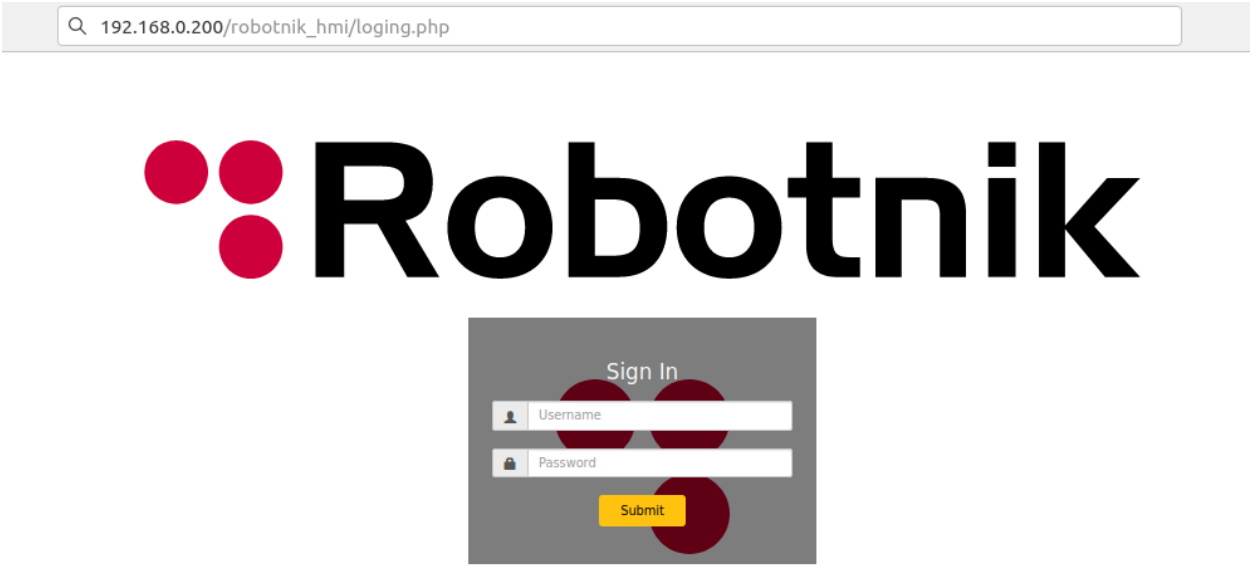


Figure 17. HMI entry page.

In order to login, use the admin credentials:

Username: admin Password: 4dm1n

Or you can use the root credentials. The root user has access to develop tabs.

Username: root Password: R0b0tn1K

It is recommended to change the password when accessed for the first time.

2.3. Interface overview

When the user is logged in, the HMI toolbar is shown on the top of the image. It is used to change between the different control tools.

Figure 18. Control tools.

By default, the Monitor tool is displayed on the interface. On the right of the toolbar, you can select the language of the web page or administrate the accounts.

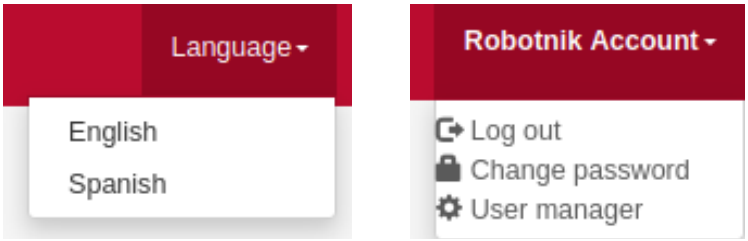


Figure 19. Left: language settings. Right: account settings

The HMI is capable of offering the following capabilities:

- **Monitor**: it is used to build maps, navigate, create points of interest, and check the status of the robot (battery, motor status, CPUs, etc).

- **Sequencer**: it is used to create sequences of actions. For example: send the robot to a specific point, go to charge, uncharge, and go to another point.

- **Scheduler**: it is used to start a mission or sequence at a specific time.

These capabilities will be explained in the following charpeters of this manual.

2.4. Monitor tool

The monitor page is the default page that appears after login. It has the following purposes:

- Visualize the robot and the environment got by the laser
- Create maps
- Localize the robot
- Navigate the robot
- Create Points of Interest (POIs)
- Monitor the status of the robot (battery, safety, sensors, motor, GPIO, and CPU)
- Check some logs

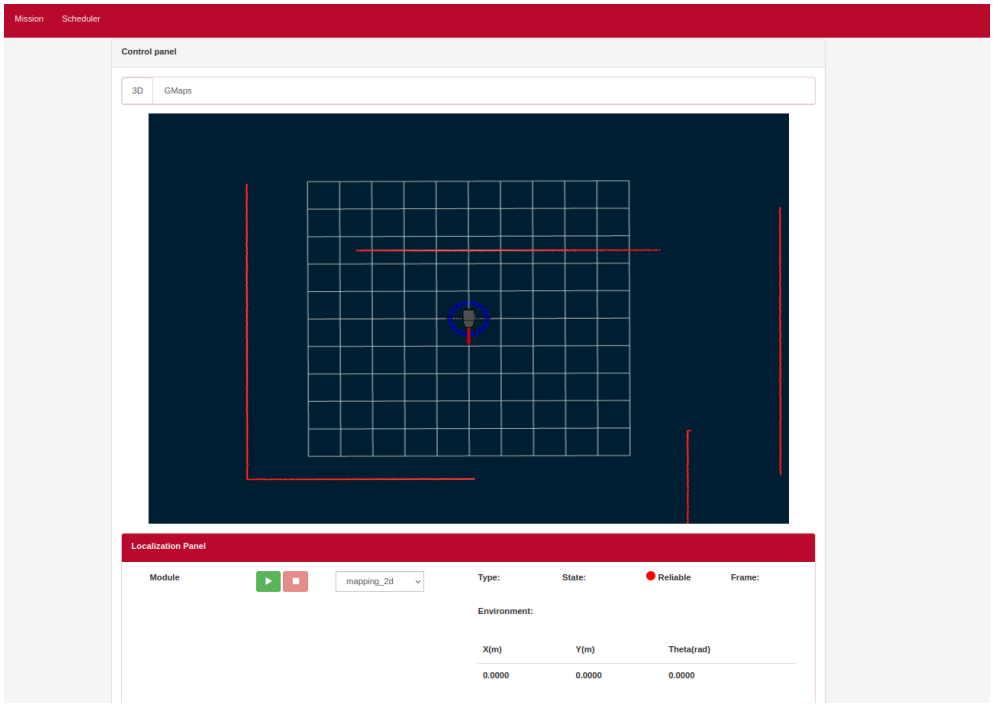


Figure 20. Monitor page.

Note: The first time the robot brings up, the robot may appear incomplete in the image. This is a usual behavior. In order to see a proper visualization, it is needed to create or load a map.

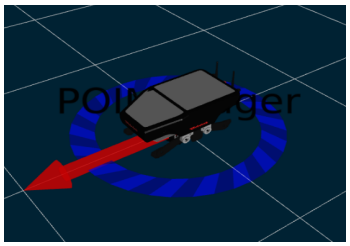


Figure 21. Initial robot visualization.

2.4.1. Control panel

This tab allows you to visualize the general status of the robot. It also allows you to create and load maps. Through the 3D viewer you can interact with the map and create points of interest and make the robot navigate through the map.

You can navigate in the 3D viewer with the mouse.

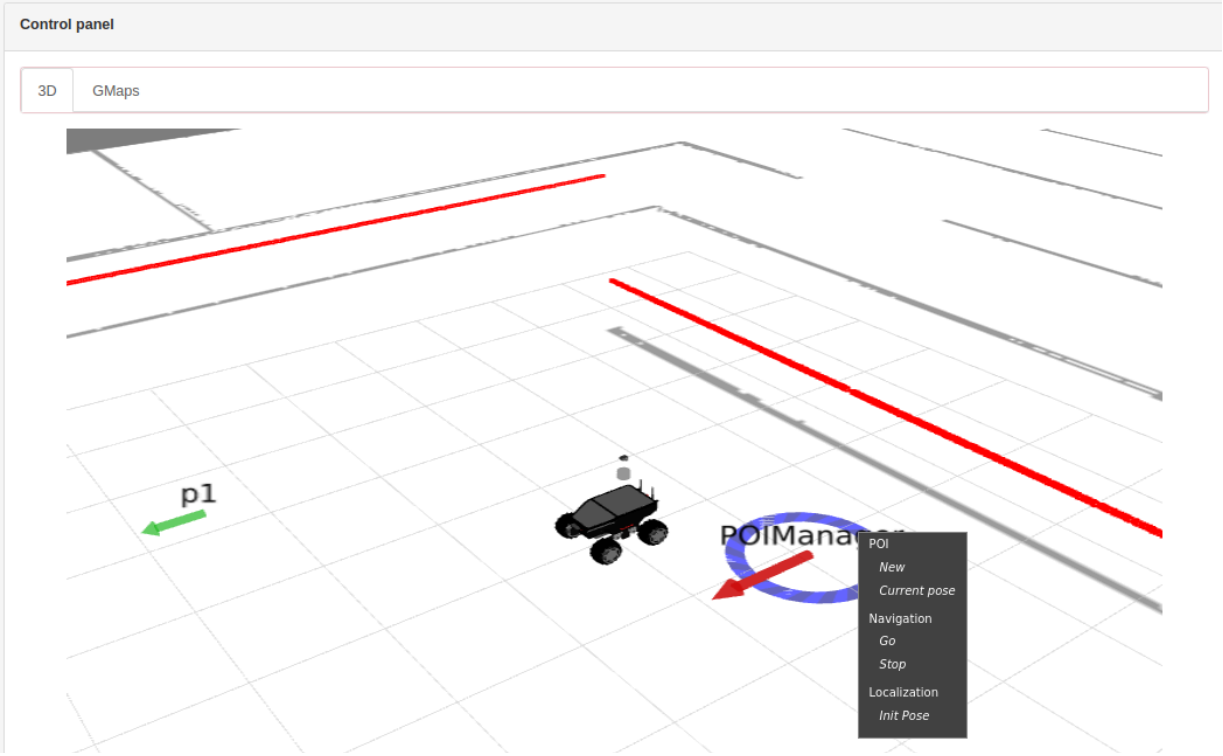


Figure 22. HMI 3D viewer.

These are the following buttons:

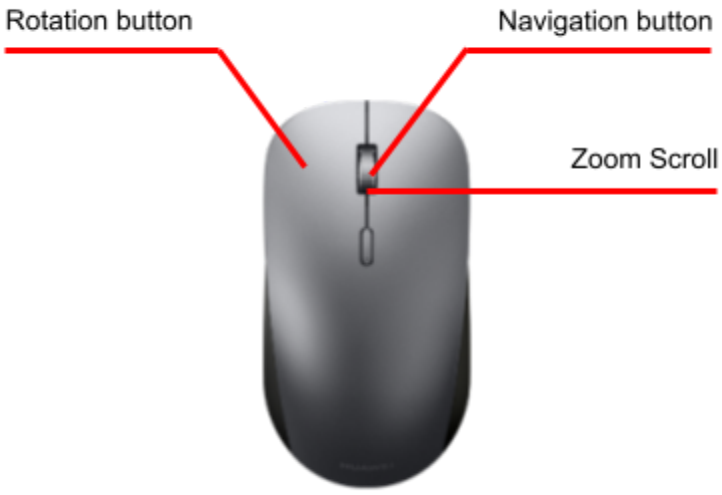


Figure 23. Mouse controls.

- **Rotation button:** hold the left mouse button to rotate in the map.
- **Navigation button:** hold the central scroll button to navigate in the map.
- **Zoom Scroll:** scroll up/down the scroll button to zoom in/out the view.

At the top of the control panel there are two tabs: 3D and GMaps.

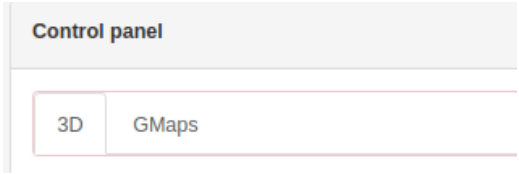


Figure 24. Control panel tabs.

- **3D:** 3D visualization of the robot in the map. In this window, the robot, the laser and the map are represented.

⚠ This window is only available if the robot has a laser sensor installed and a map loaded.

- **GMaps**: GPS representation of the robot in the world using Google Maps.

⚠ This window is only available if the robot has a GPS installed and has an internet connection.

In the 3D visualization tab, it appears the POIManager cursor. This is an interactive marker used for the localization and navigation.

Figure 25. POI manager cursor.

Move the POIManager cursor by holding the red arrow, rotate it by holding the blue circle.

Figure 26. Left: Move POIManager. Right: Rotate POIManager.

With a right click on the POIManager marker, a tab with different option will appear:

Figure 27. POIManager options.

- POI
  - ***New***: save a new point in the current position of the POIManager cursor in the map.
  - ***Current pose***: save a new point in the position of the robot in the map.
- Navigation
  - ***Go***: send the robot to a specific point. It can be done using the POIManager cursor or selecting a POI.
  - ***Stop***: stop the robot while it is navigating to a point.
- Localization
  - ***Initpose***: set the current position of the POIManager cursor, as the initial position of the robot in the map. This position is useful to the localization algorithm, and can be set manually at the beginning of the localization process.

2.4.2. Localization panel

The localization panel is responsible for starting and stopping the mapping, localization and navigation processes. It also monitors these processes.

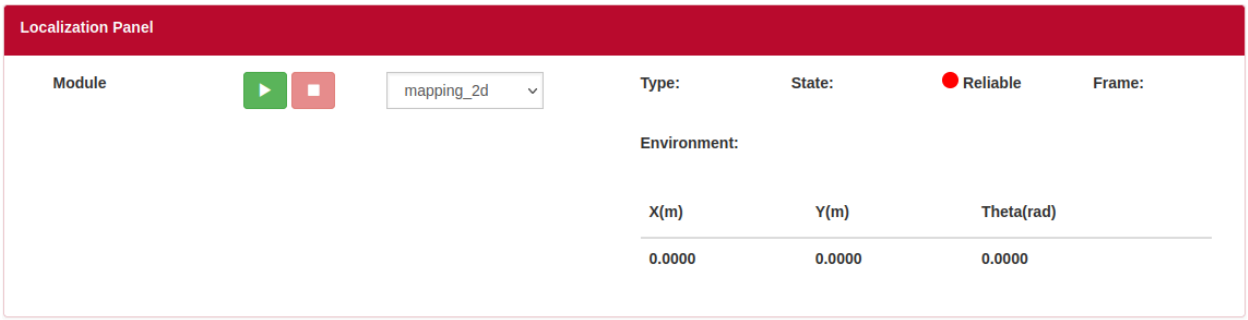


Figure 28. Localization panel.

In order to create a map or start the localization and navigation, you can use the drop-down menu placed in the left side of the localization panel to load a specific module.

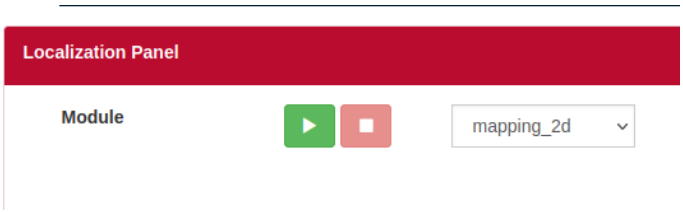


Figure 29. Localization panel modules.

There are four available modules:

- **mapping_2d:** it is used to create 2D maps.
- **localization_2d:** it is used to localize the robot in a 2D map. Navigation is also launched in this module.
- **mapping_3d:** it is used to create 3D maps.
- **localization_3d:** it is used to localize the robot in a 3D map. Navigation is also launched in this module.

⚠ The mapping_3d and localization_3d are extra modules that must be purchased separately, for that reason they are not available in all robots.

Select a module in the drop-down menu. Depending on the selected module, new parameters will be enabled in this section.

Figure 30. Localization panel menu.

Press the play button to start the module on the robot. On the right side of the HMI, an status message will be displayed.

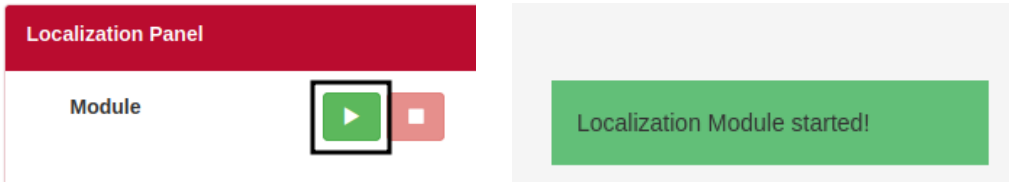


Figure 31. Localization panel run button.

In order to stop the module, press the stop button. On the right side of the HMI, a message will be displayed.

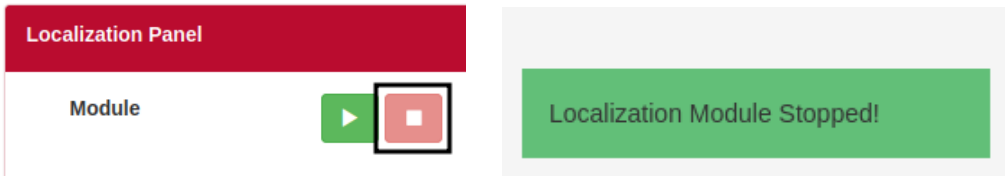


Figure 32. Localization panel stop button.

⚠ Before stopping the module, make sure that the map is saved or the robot is in a safe zone.

2.4.2.1. Mapping module

When the mapping module is selected, the robot will start to build a map using the laser readings.

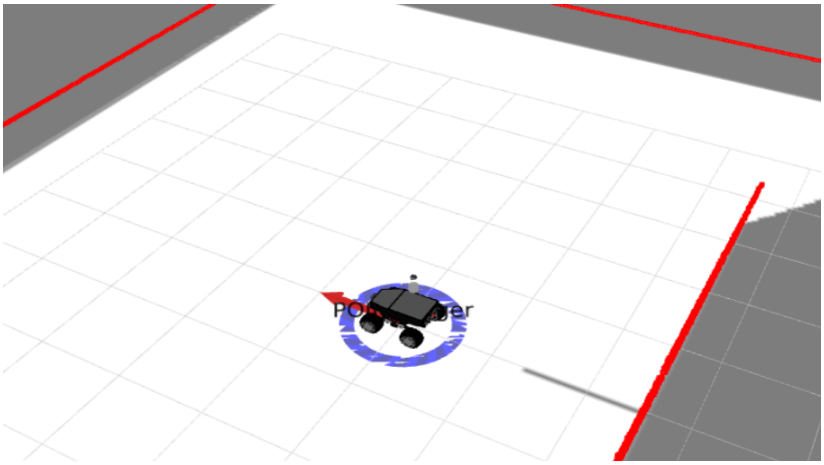


Figure 33. Localization panel robot mapping.

The **Save Map** tool will be displayed on the localization panel.

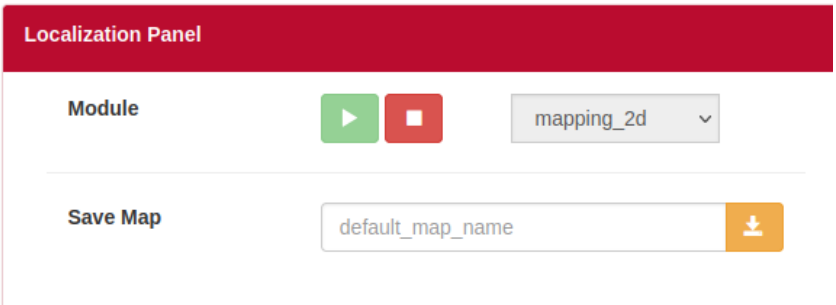


Figure 34. Localization panel save map tool.

Once the robot has built the map, introduce the name of the map in the text box, then save the map by clicking on the yellow icon.

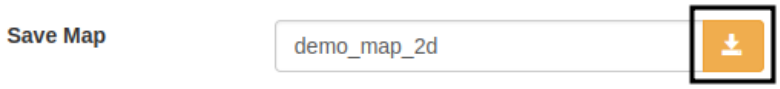


Figure 35. Localization panel save map name.

On the right side of the HMI, a message will be displayed.

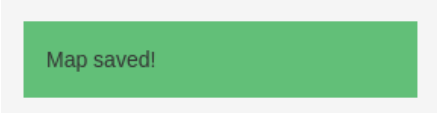


Figure 36. Localization panel map saved pop-up.

Finally, stop the mapping module in order to load another module.

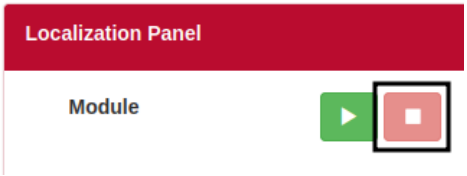


Figure 37. Localization panel stop button.

2.4.2.2. Localization module

When the localization module is selected, the **robot will try to localize and navigate in a map**. Depending on the configuration, a default map can be loaded or not. In any case, the desired map can be selected.

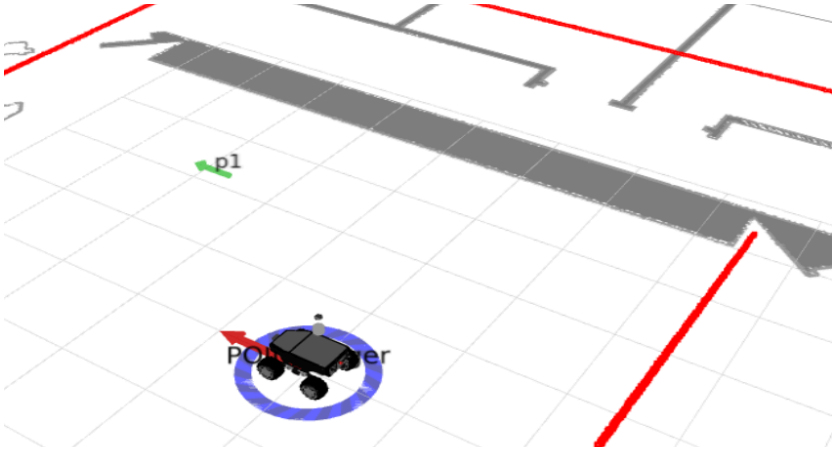


Figure 38. Localization panel robot localization.

The Environments tool will be displayed on the localization panel.

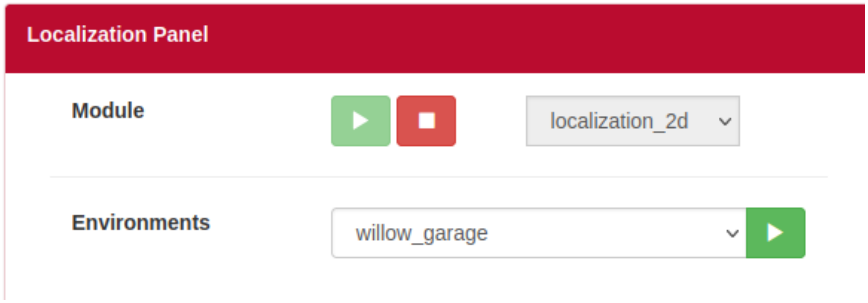


Figure 39. Localization panel environments tool.

In order to select a map, click on the drop-down menu.

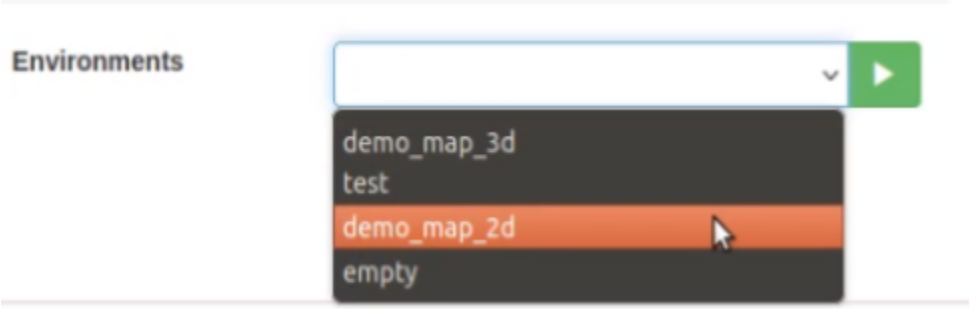


Figure 40. Localization panel environments menu.

Load the map by clicking on the green button.

Figure 41. Localization panel run localization button.

On the right side of the HMI, a message will be displayed.

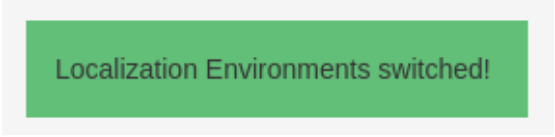


Figure 42. Localization panel localization switched pop-up.

When the map starts, if the robot position matches with the origin of the map, the robot will be localized. The laser readings match with the map. On the 3D localization, if the 3D map and environment are good, the robot can localize automatically even if the initial position is wrong. In any other case, the localization algorithm will need an initial pose reference.

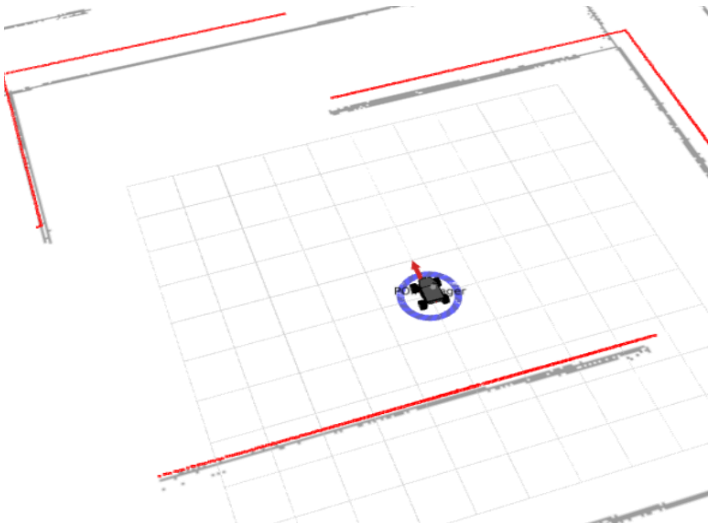


Figure 43. Robot localized.

In most cases, the robot position does not match with the origin of the map. The laser readings do not match with the map For that reason the localization algorithm will need an initial pose reference.

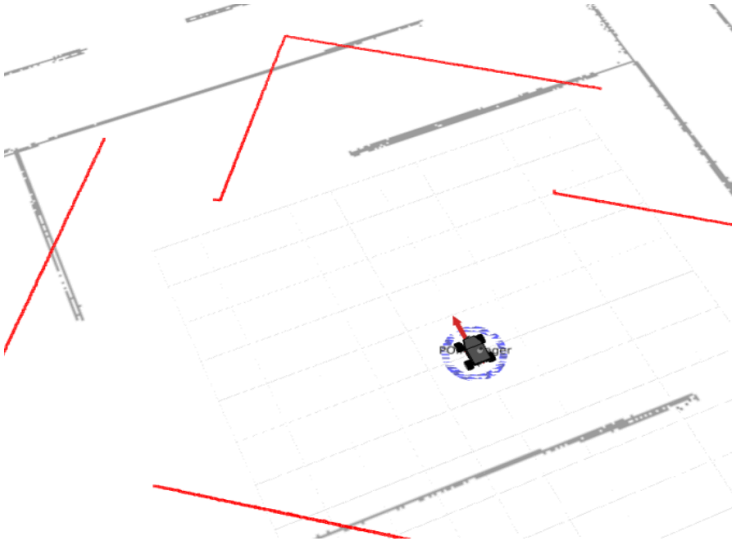


Figure 44. Robot not localized.

In order to set an initial pose from the Control Panel, move the POIManager cursor to the real position of the robot in the map.

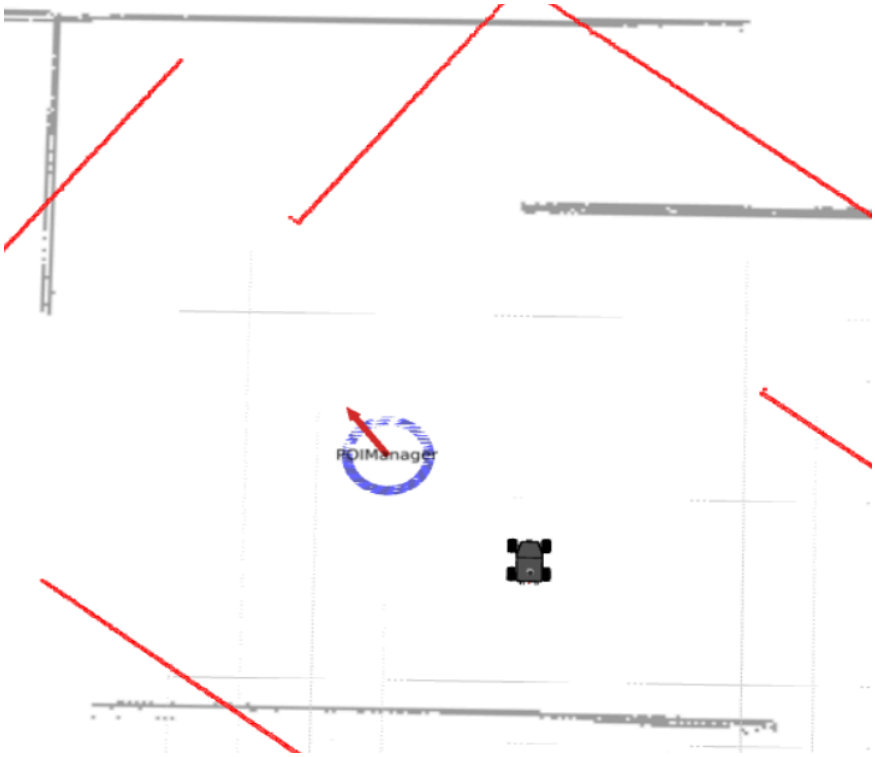


Figure 45. Robot not localized, it needs an initial position.

Then, click on the right button of the mouse and select Init Pose

Figure 46. Robot initial position

The robot will be localized at the position of the POIManager cursor.

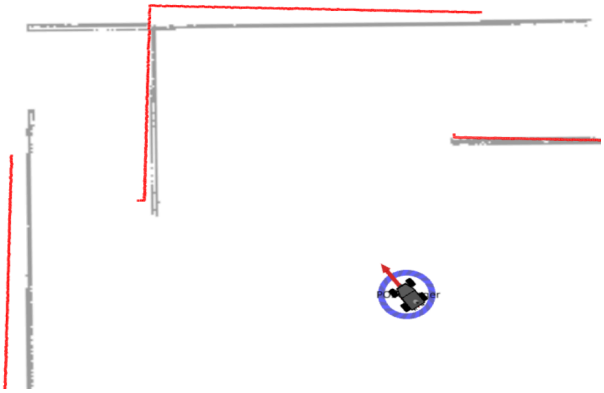


Figure 47. Robot almost localized

Do not worry if the laser readings do not match exactly with the map, the localization algorithm is able to rectify these small errors. Move the robot in order to rectify the error.

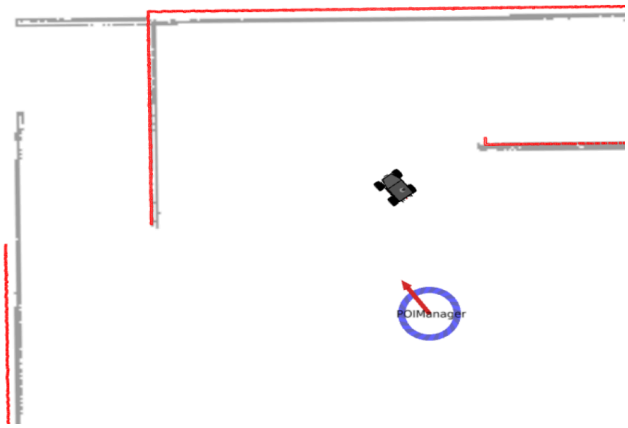


Figure 48. Robot well localized

Finally, stop the localization module in order to stop the localization and navigation.

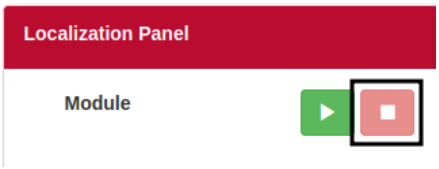


Figure 49. Localization panel stop localization button.

WARNING

Be careful when stopping the localization. If the localization is off, the navigation will not work. The robot could be navigating or executing a critical task.

2.4.2.3. Navigation module

The navigation module is launched automatically when the localization module starts. Please, make sure that the robot is localized before following the next steps.

WARNING

Make sure that the robot is localized. Otherwise the robot can move erratically and can collide with the environment if there are obstacles non detectable by the sensors. Also, the navigation points and the saved points in the map will not correspond with the real localizations.

In the Control Panel, the navigation can be controlled using the POIManager cursor. Place the POIManager in the desired position where the robot will navigate. Then click on the right button of the mouse to view the options.

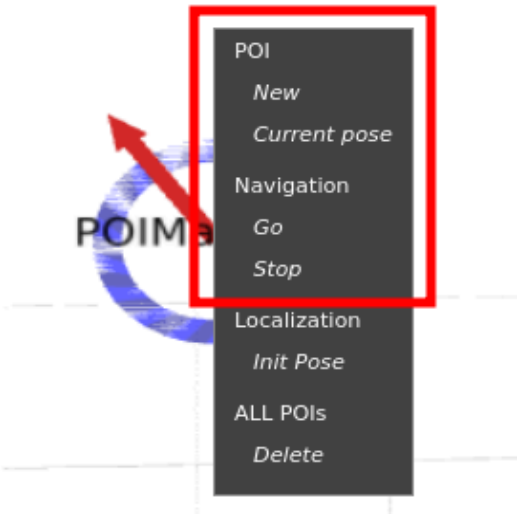


Figure 50. POIManager options for POIs.

In order to navigate, select the ***Go* option**. The robot will navigate to the POIManager cursor position.

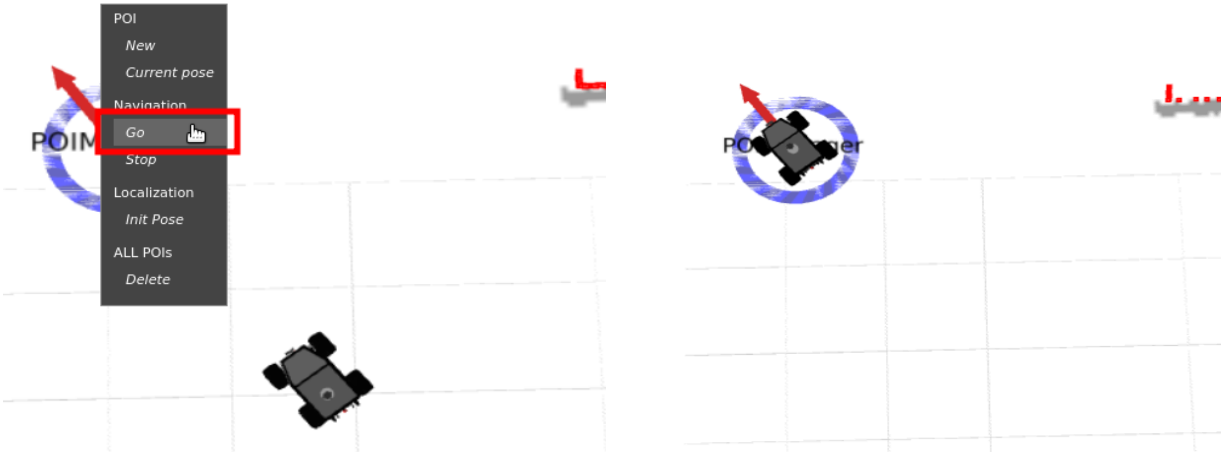


Figure 51. Left: Setting a navigation goal. Right: Robot reaches the goal.

In order to stop the navigation, select the ***Stop* option**. The goal will be canceled and the robot will stop immediately.

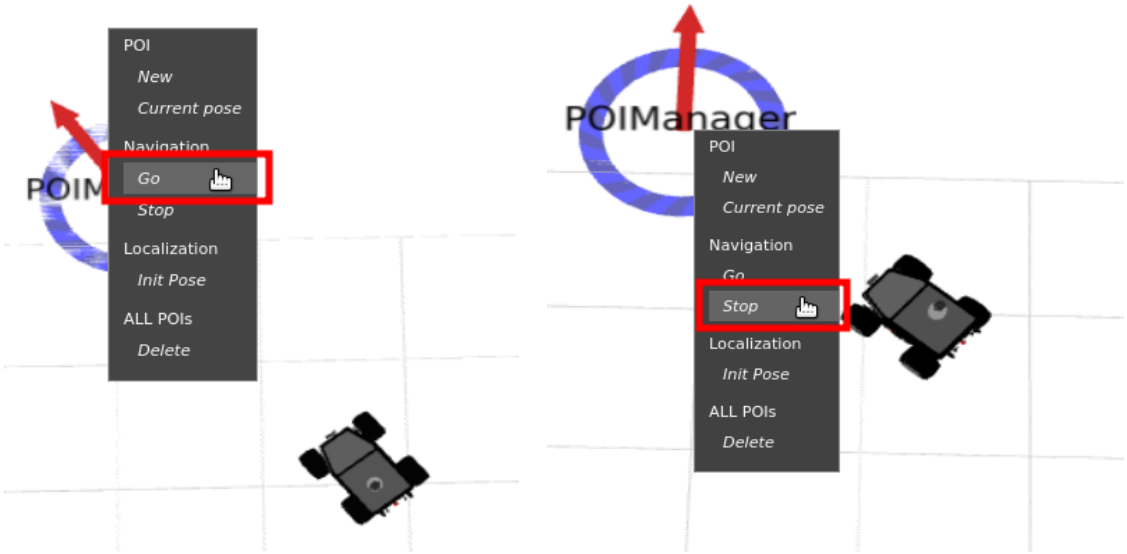


Figure 52. Left: Setting a navigation goal. Right: Robot stops immediately.

In order to save points, select the ***New* option**. A point will be saved in the POIManager cursor position.

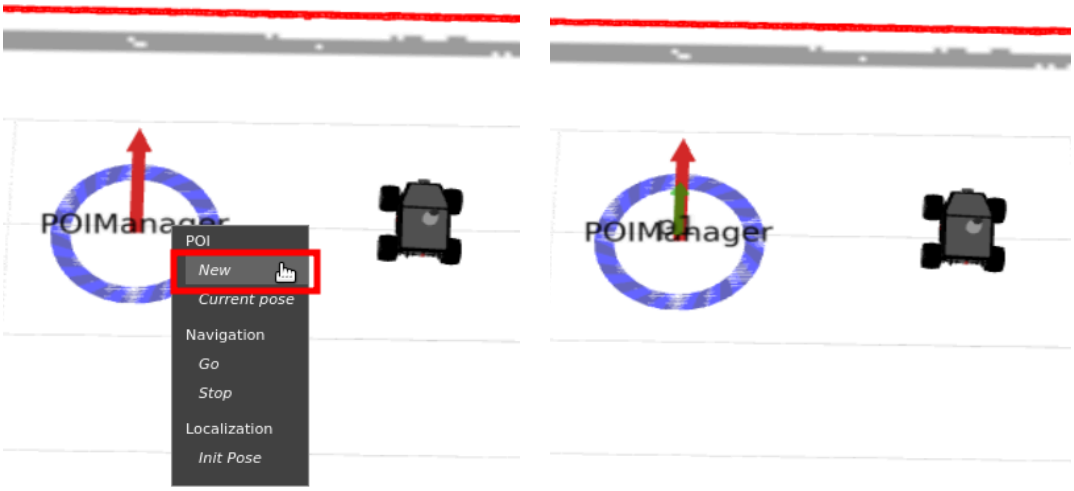


Figure 53. Left: Creating a POI. RIght: POI created in the POIManager position.

Another way to save points is selecting the ***Current pose* option**. A point will be saved in the robot position.

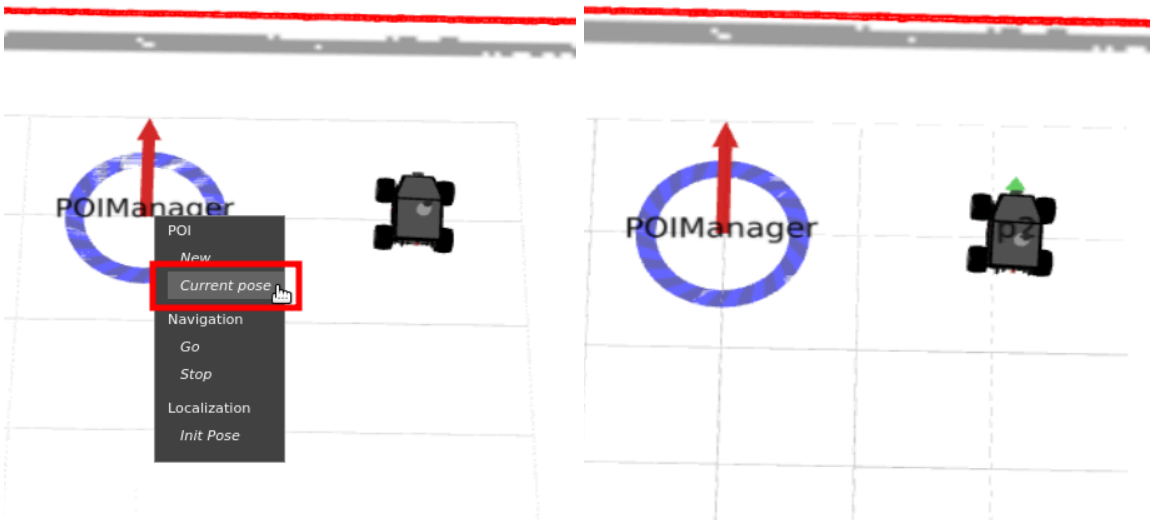


Figure 54. Left: Creating a POI. Right: POI created in the robot position.

The saved points can be used as navigation goals and can be edited. Click on the right button of the mouse to view the options.

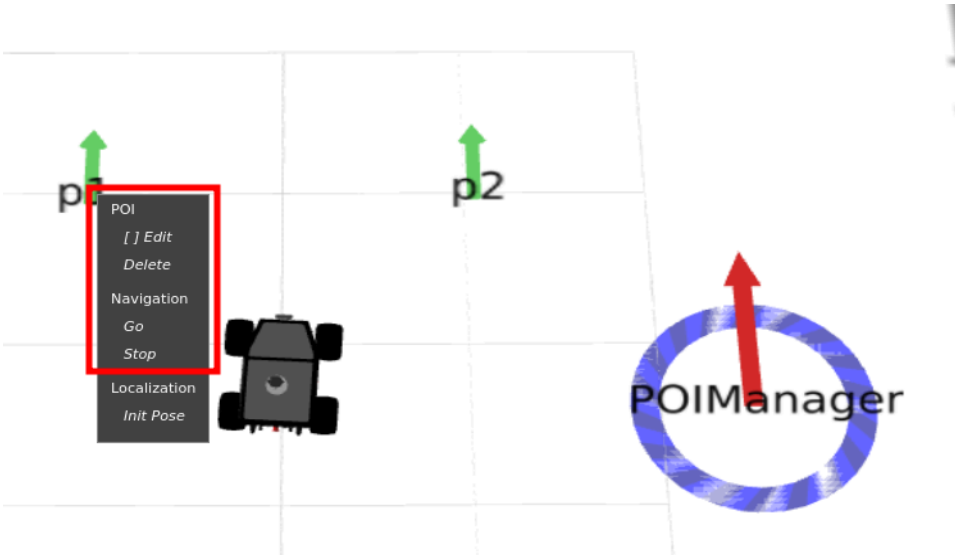


Figure 55. POI options.

In order to modify the saved point, select the ***Edit option.***

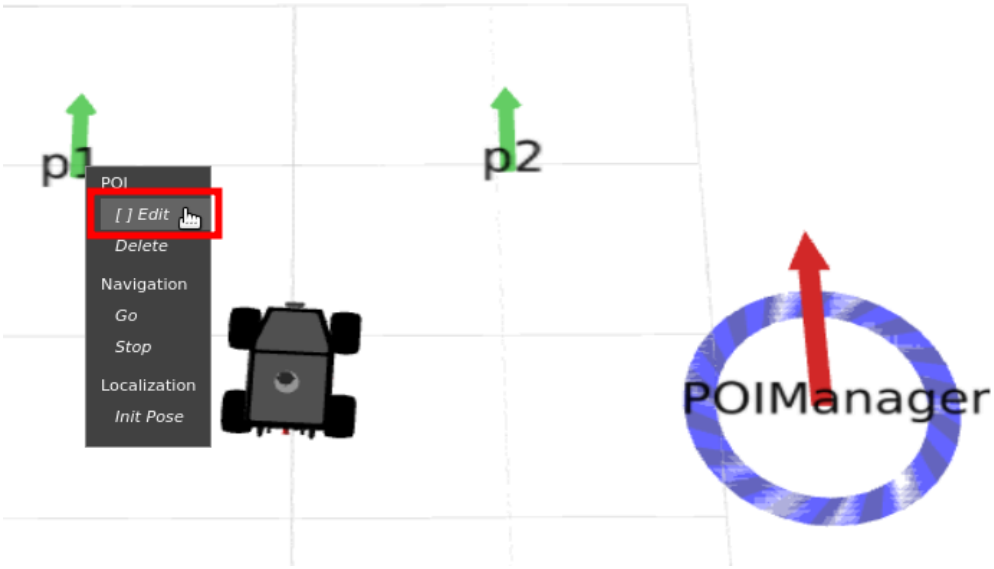


Figure 56. Edit POI option.

Then move the point like a POIManager cursor.

Figure 57. Left: POI ready to edit. RIght: POI edited in another position.

In order to remove a saved point, select the ***Delete option***.

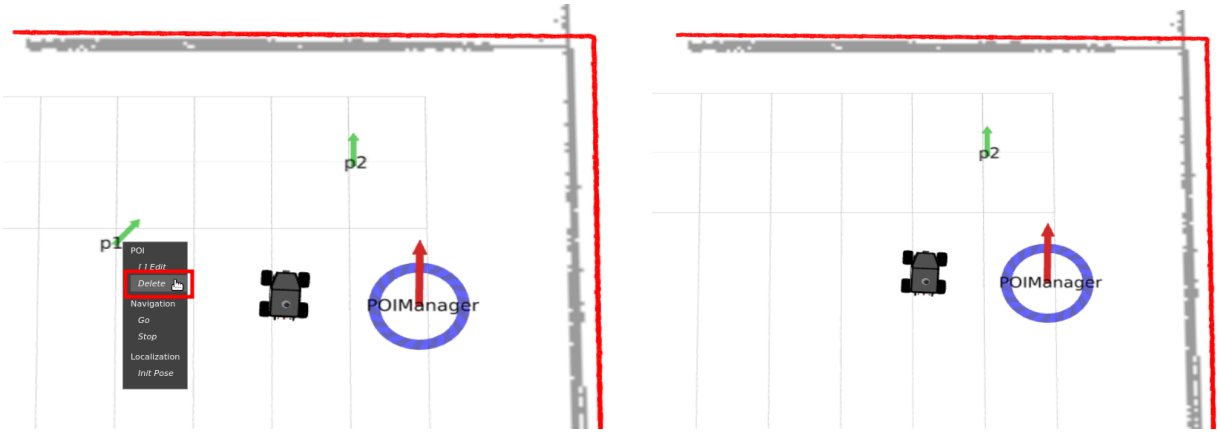


Figure 58. Left: Delete POI option. Right: POI deleted.

2.4.2.4. Status panel

The status panel shows information about the state of the modules (mapping_2d, mapping_3d, localization_2d, localization_3d, etc) and the current robot position.

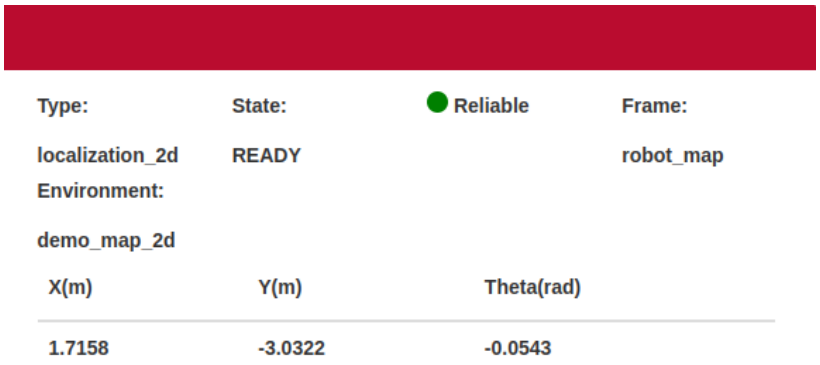


Figure 59. Status panel.

The following states can be monitorized:

- **Type**: Current module loaded: mapping_2d, mapping_3d, localization_2d, localization_3d and so on.

- **State**: State of the current module loaded. There are different states:

    - **STANDBY**: The module is starting, the robot can not localize and navigate yet.
    - **READY**: The module is working, the robot will localize and navigate.
    - **EMERGENCY**: The module is not working due to a sensor failure, motor failure or crash localization/navigation algorithm.
    - **VOID**: if there is not any state it means that the module is not loaded.

- **Reliable**: Indicator that shows if the robot position is reliable. There are two colors:

    - **Red**: The robot position is not reliable. Set an initial pose with the POIManager cursor in order to get a reference for the localization algorithm.
    - **Green**: The robot position is reliable. It means that the initial pose was done with the POIManager, but it does not mean that the robot is well localized.

WARNING

The reliable indicator turns green when the initial pose is set with the POIManager cursor, but it does not mean that the robot is well localized. Please, check that the laser scans match with the map to make sure that the robot is localized.

- **Frame**: Name of the frame where the map is attached. By default it is *robot_map*.
- **Environment**: Map currently loaded for the localization module.
- **X/Y/Theta**: Current robot position in the map. These values are only reliable if the robot is well localized, it means that the laser scans match with the map.

Finally, if there is not any module loaded, the status panel will be void.

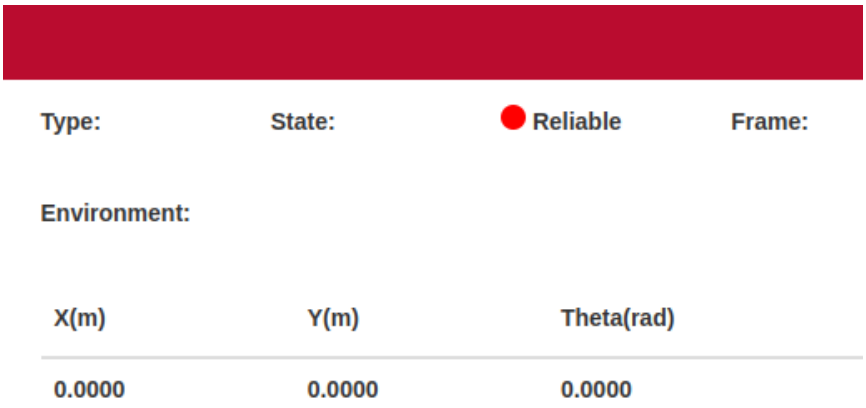


Figure 60. Void status panel.

2.4.3. Logging console

The logging console shows the feedback about different commands executed in the robot. It is useful for debugging when some component is not working as expected.

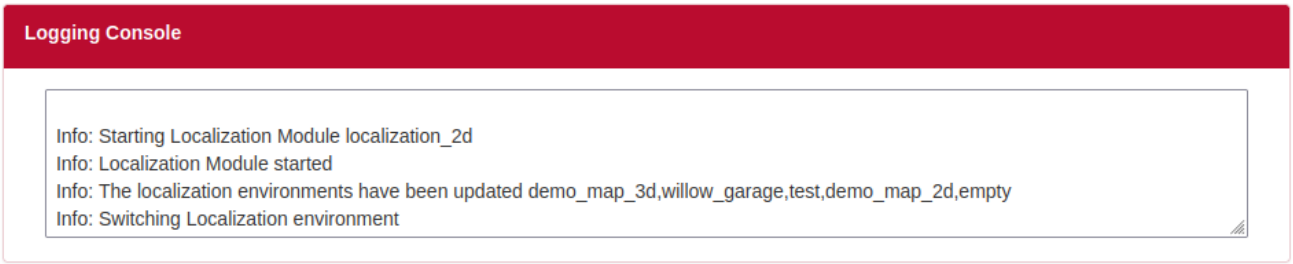


Figure 61. Logging console

If there is any error with the modules, the logging console will show it:

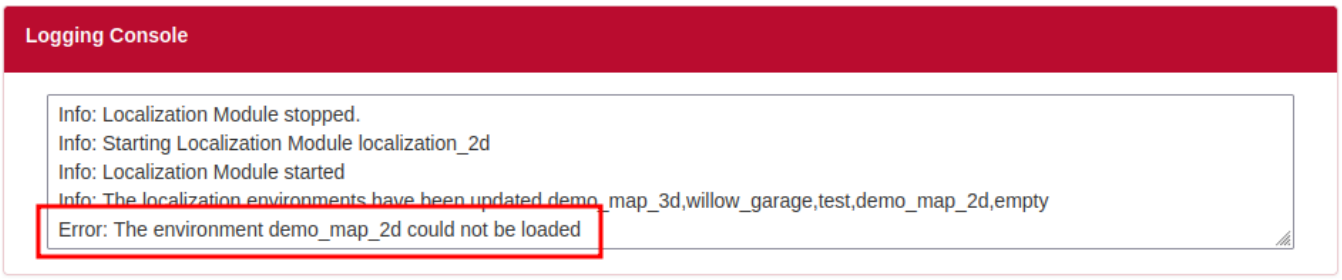


Figure 62. Logging console output

Finally, the console can be resized from the bottom right corner.

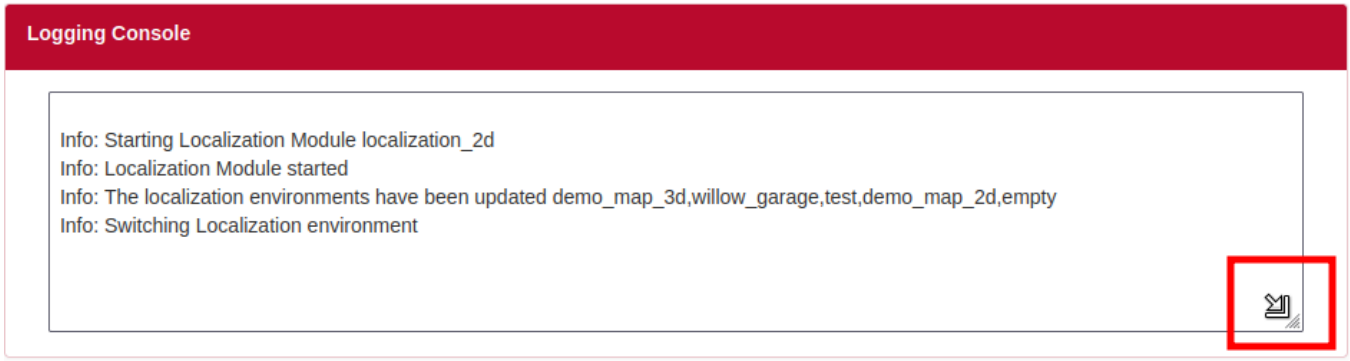


Figure 63. Logging console resize

In summary, the logging console is capable of showing the following information:

- Commands executed in the robot from the HMI
- Status of the modules

2.4.4. Robot status

In the status summary section, on the one hand you can see the status of the robot, where the status of the different components of the robot (battery, motors, sensors, elevator, etc.) is grouped, and on the other hand you can see the navigation status, where you can see the position of the robot with respect to the fixed coordinate system and the speed.

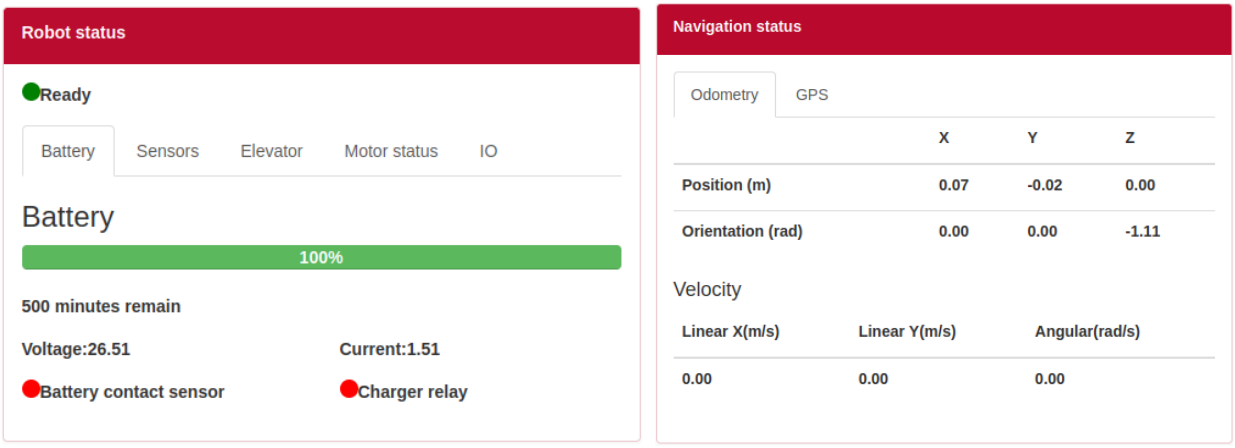


Figure 64. Robot status

2.5. Sequencer

In the sequencer tab you can create new sequences or select (via the drop-down menu) already created sequences to run, modify or delete them.

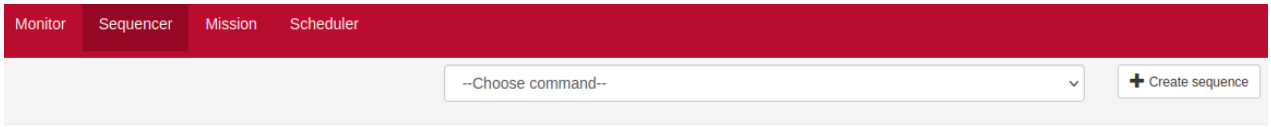


Figure 65. Sequencer.

If "Create Sequence" is selected, a new view will appear with all available commands for the robot. Each time a command is selected, a new form will appear with the arguments available for that command. Once the desired sequence has been created, name it using the text box and press the Save button.

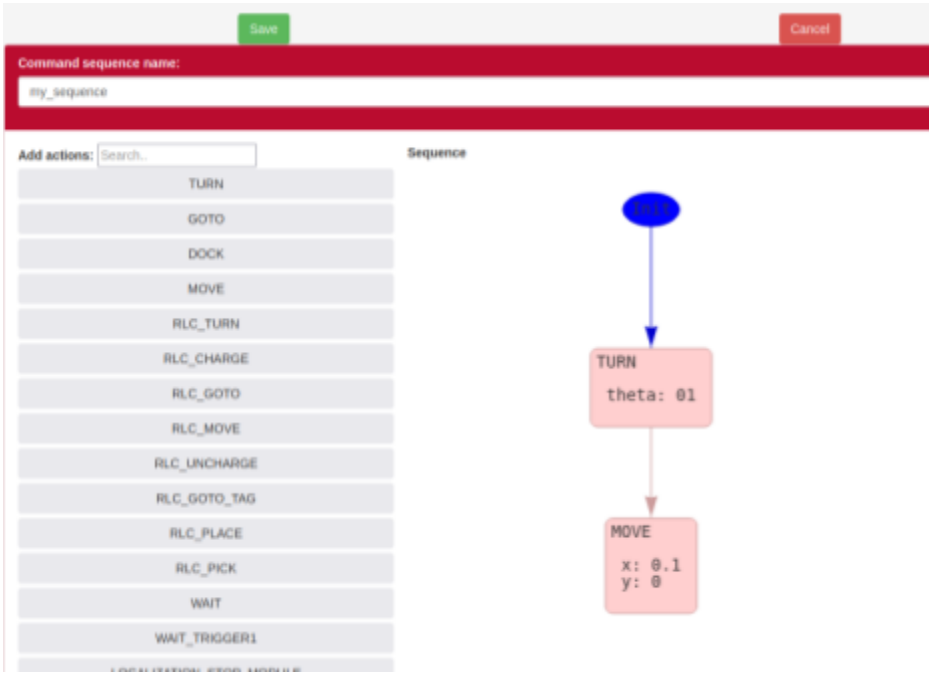


Figure 66. Sequence 1.

If you select a sequence that has already been created, a view will appear with the list of commands that make up the sequence. You will be able to modify, delete or execute it.

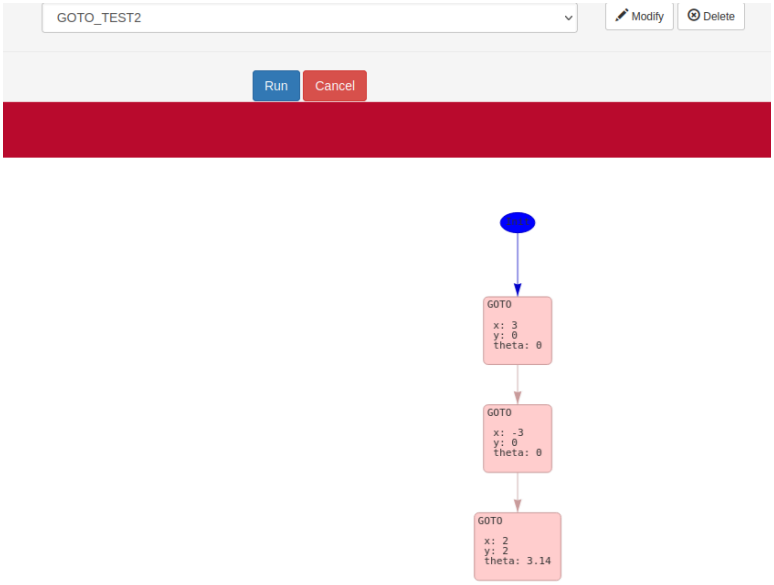


Figure 67. Sequence 2.

If you modify an existing sequence, a view similar to the Create sequence will appear.

2.6. Mission

This tab allows the creation of missions, which are a set of sequences. A mission allows configuring different sets of sequences for different robot states (low battery, error or normal state).

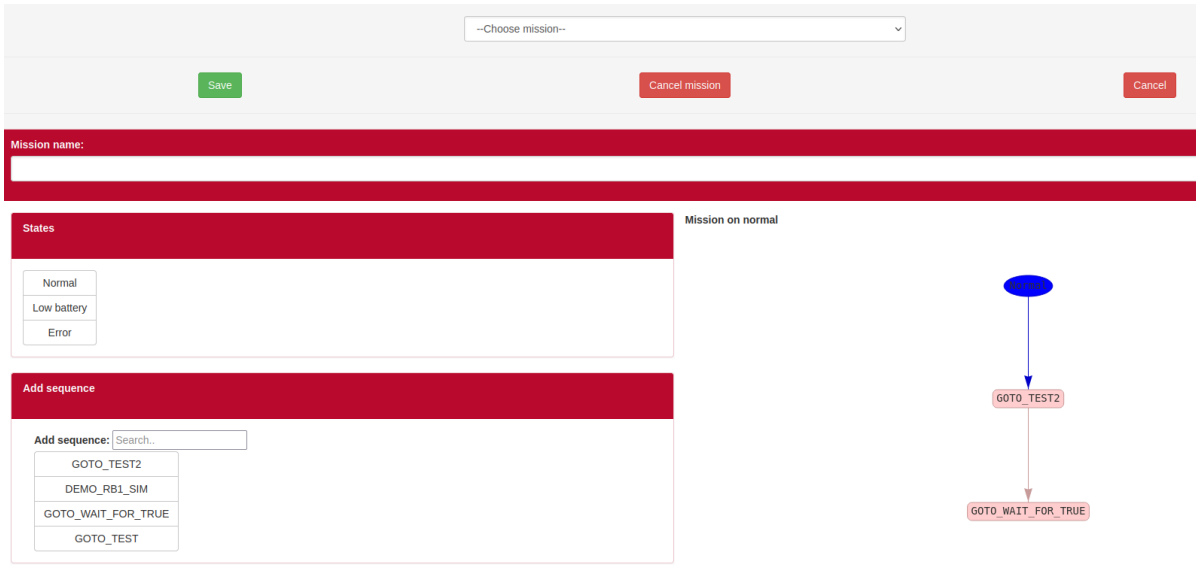


Figure 68. Mission.

2.7. Scheduler

This tab allows you to schedule missions. Pressing the "+ New event" button displays a form to configure the scheduling of a new mission.

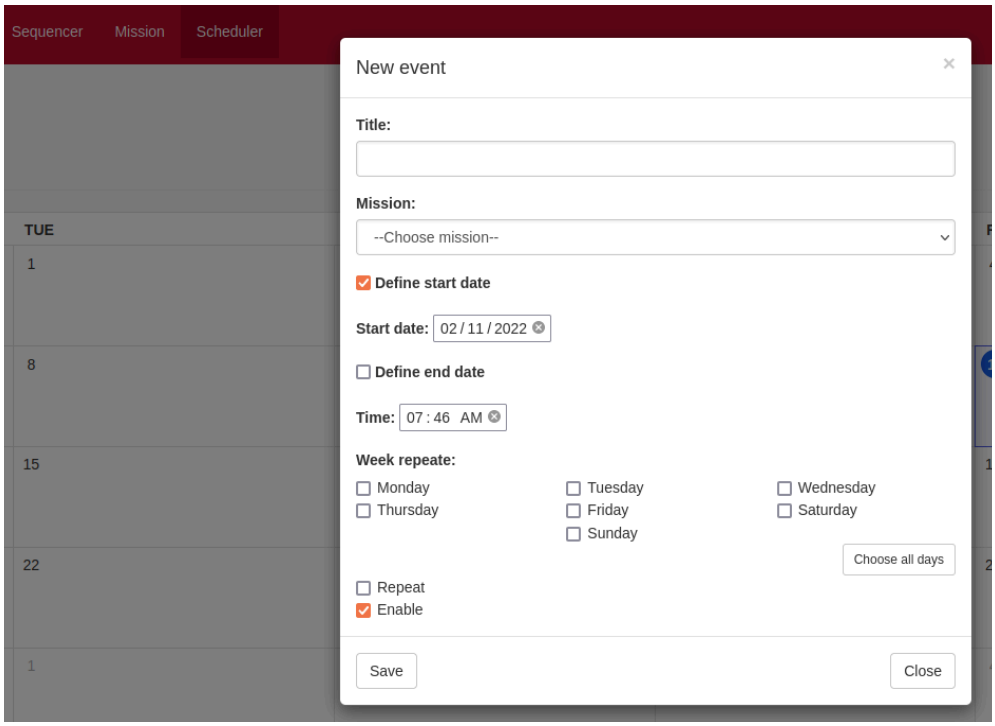


Figure 69. Scheduler.

3. Robotnik Integration URCap

The Robotnik Integration URCap is only available for the robots that integrate a Universal Robots arm. The UR user should be familiar with the UR and Polyscope environment.
For further information about general UR topics, please consult UR official documentation.

Robotnik Integration URCap is a tool to control both the arm and the mobile base, performing movements and navigation actions in a simpler way. The URCap is an internal application running locally on the UR control box.

3.1. Installation parameters

Once the URCap is installed, go to the Installation section to configure the robot. Put the IP address and the port of the server of the URCap (located in the Robotnik mobile robot). When the arm can communicate with the base, the green check icon will appear.

Figure 70. Installation parameters of Robotik Automation URCap.

3.2. Program nodes

The URCap includes a set of program nodes that allows the interaction with the Robotnik's mobile base.

All program nodes check connection between the UR and the Robotnik's mobile base and show a red label in the command tab.

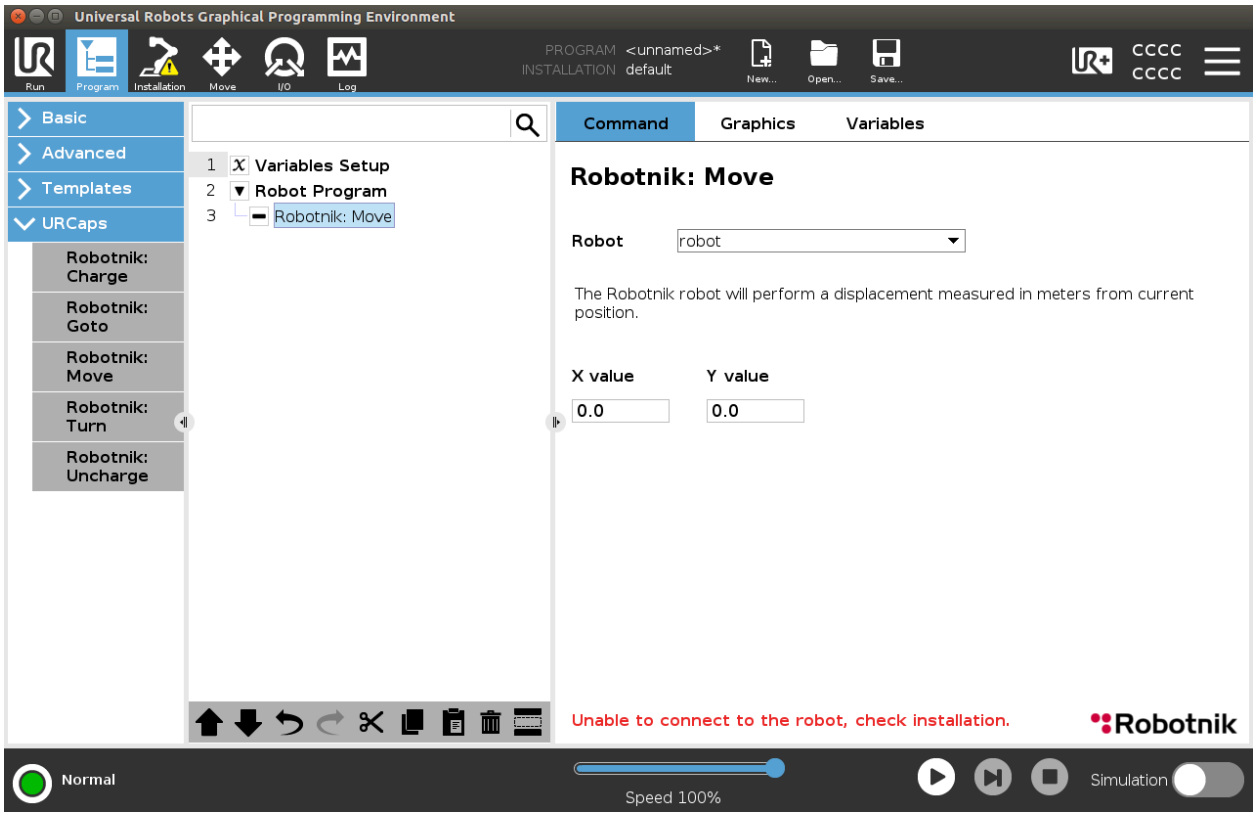


Figure 71. Program nodes.

3.2.1. GoTo

The GoTo program node makes the robot navigate to a certain pose defined in a map in a safe way. The different tags are defined in the map displayed in Robotnik's HMI (see section 1).

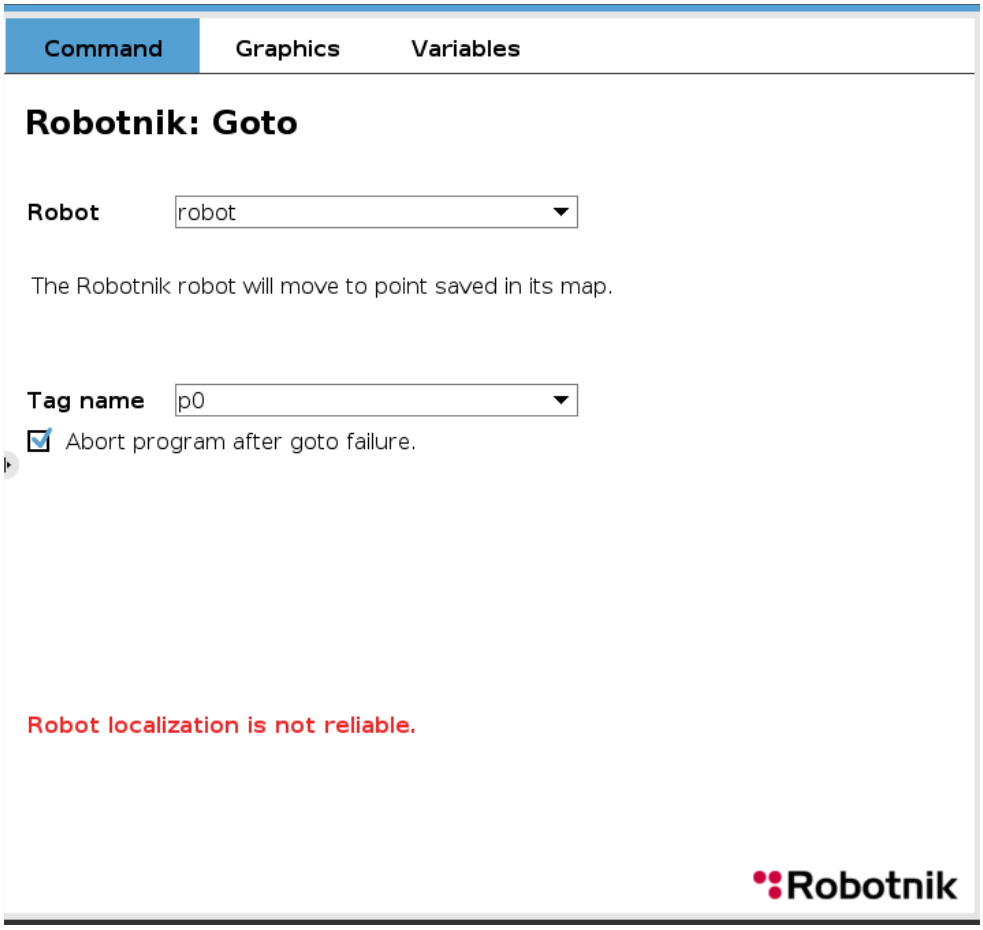


Figure 72. GoTo node

You can use the checkbox to select if you want the robot to abort the program after command error or continue the program. This node will check for the localization to be running and being reliable to set the defined state. If the defined state is set to false, the node will be yellow and the UR program will not be executed.

3.2.2. Charge

The UR Program can be also combined with a charging procedure of the mobile platform.
The Charging station is an optional accessory of the Robotnik robots. Each platform is equipped with a recognizable and non- interchangeable charging station.

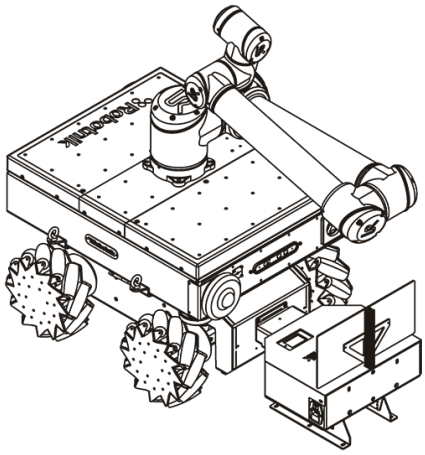


Figure 73. Charging station

For further information please consult the Charging station manual.

With Charge action the robot will go to its charging station identified by a QR code and start charging its battery.

The steps of the charging procedure are the following:

1. The robot recognizes the QR code in the charging station. For this reason to execute this structure the robot must be in front of the station. It is recommended that the robot always has a GoTo action to a point where it sees the QR code before this execution.
2. The robot will approach and make contact with the charging station.

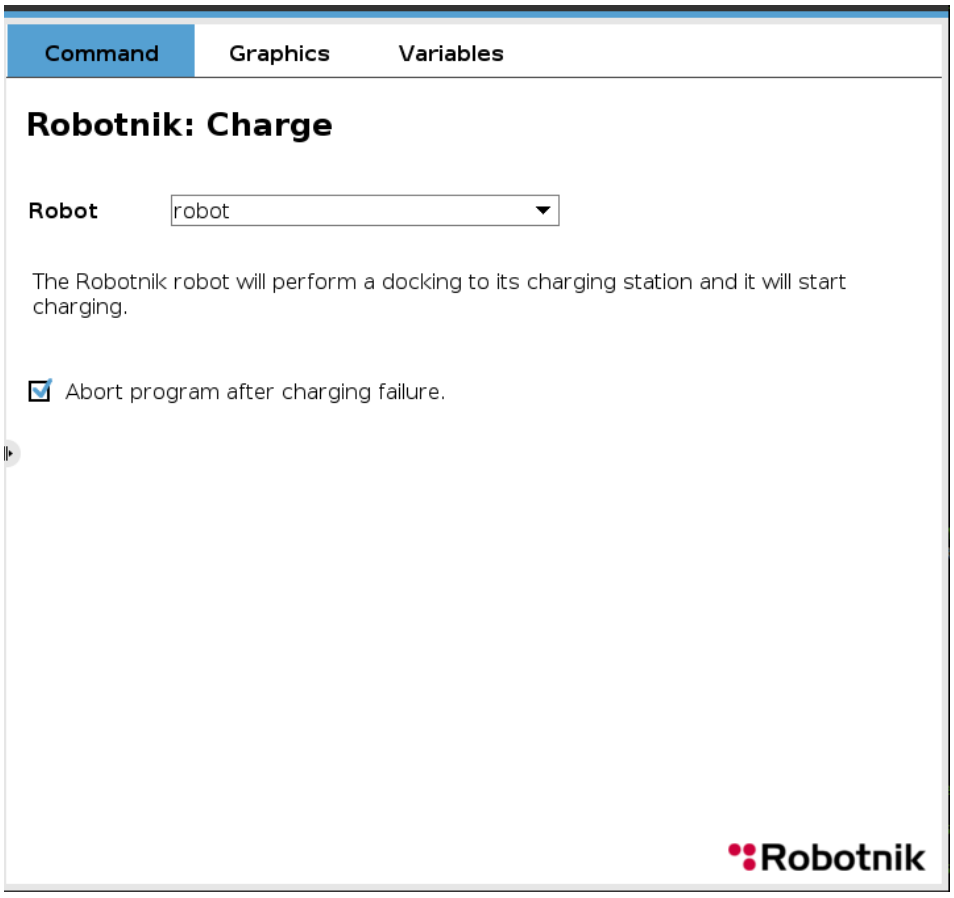


Figure 74. Charge node

You can use the checkbox to select if you want the robot to abort the program after command error or continue the program.

3.2.3. Uncharge

This action is used to properly separate the mobile platform from the charging station or docking in a safe way.

It should be always used for this purpose, this means, after a Charge action, the next movement of the mobile platform should be this one.

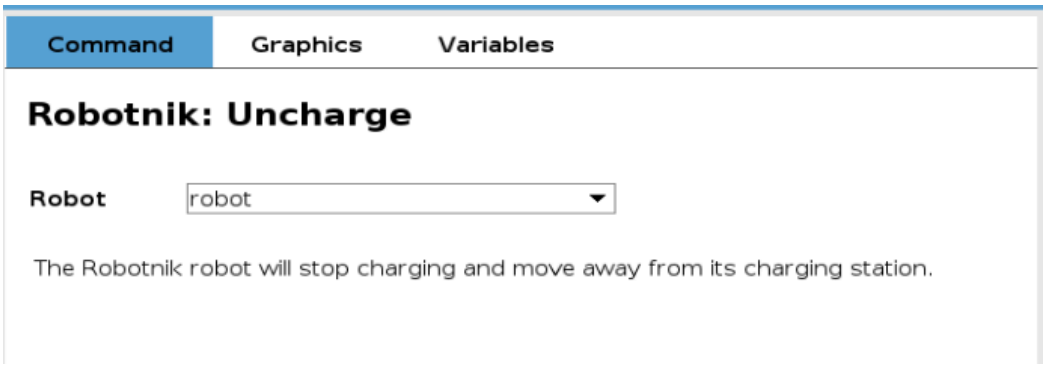


Figure 75. Uncharge node

3.2.4. Move

This structure is used to move the mobile base relatively to its current position. This block moves the platform linearly in the X or Y direction. If the robot is omnidirectional, it will be capable of performing movements on the Y axis.

The distance to be moved in the action is to be specified in meters.

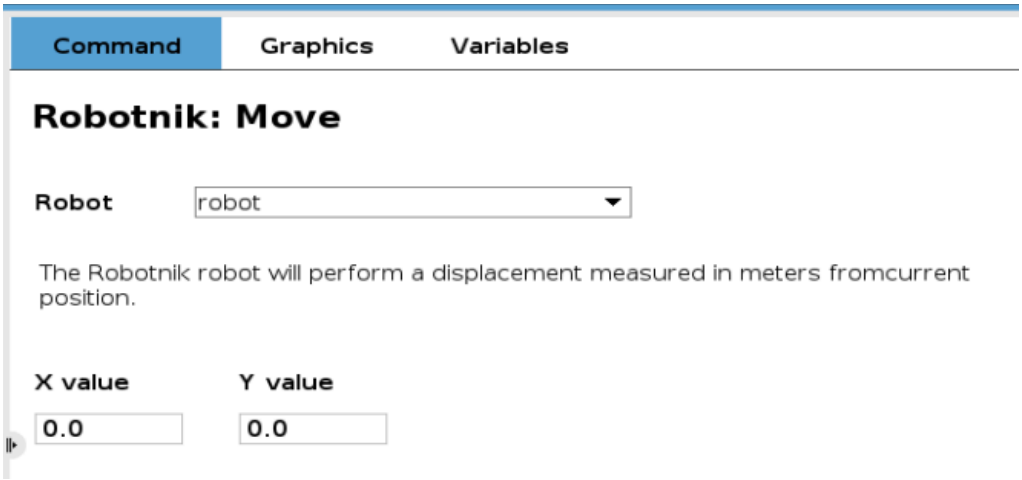


Figure 76. Move node

3.2.5. Turn

This block turns the mobile base relatively to its current position. The rotation quantity must be specified in radians by the user.

Please take into account the type of movements that are recommended for your specific robot, in order to perform the movements in a proper way and to extend the life of the robot components.

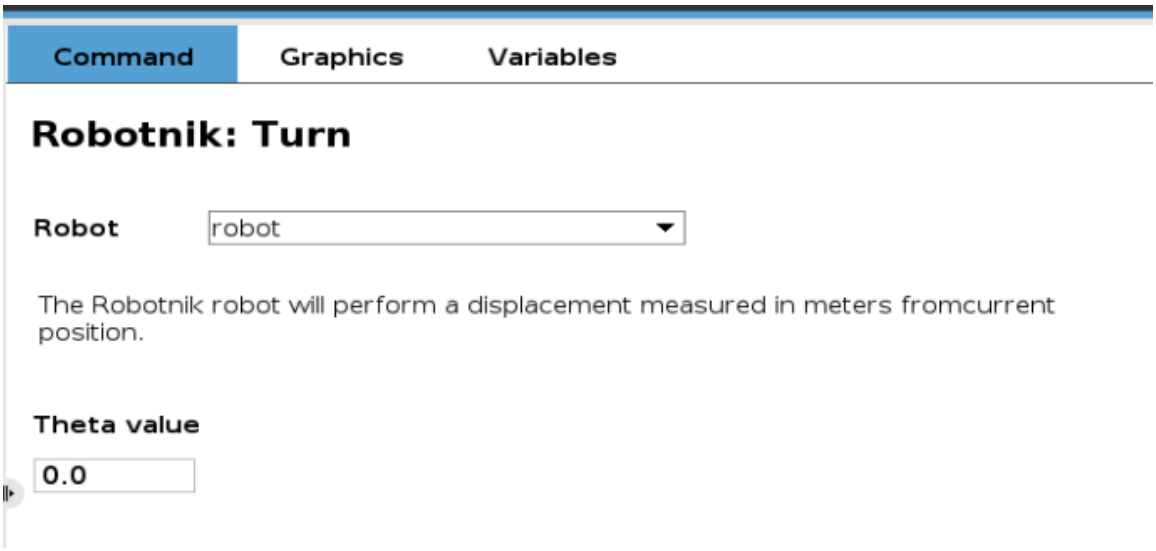


Figure 77. Turn node

3.3. Script functions

All program nodes have also been defined as URScript functions so that the user can integrate the interaction with the Robotnik base within his URScript programs.

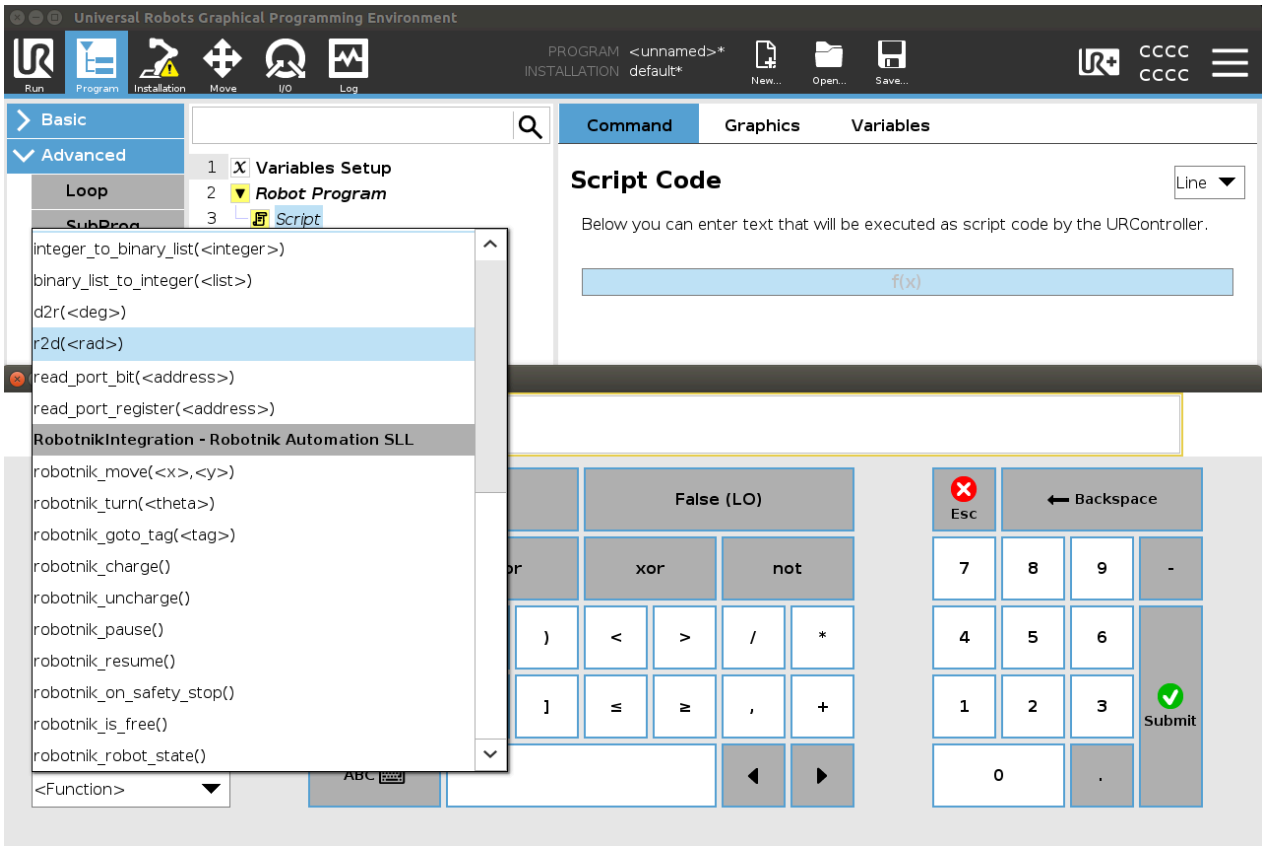


Figure 78. Script functions

There are also URScript functions with the same functionality as the program nodes explained before. Additional URScript functions are explained in the API manual.

3.4. Toolbar

The URCap enables a Toolbar where a simplified status of the Robotnik base can be visualized.

Battery level: Battery percentage of the robot.

- **Consumption**: Instantaneous current consumed by the robot
- **Charging**: Flag that indicates if the robot is charging
- **Safety mode**: Shows if the safety is enabled or not. It is set to “unknown” value when the robot does not have a safety module.

- **Environment**: Name of the map that the robot is currently using.
- **Last command**: Last command sent to the robot.
- **Command running**: Flag to indicate if the robot is running a command

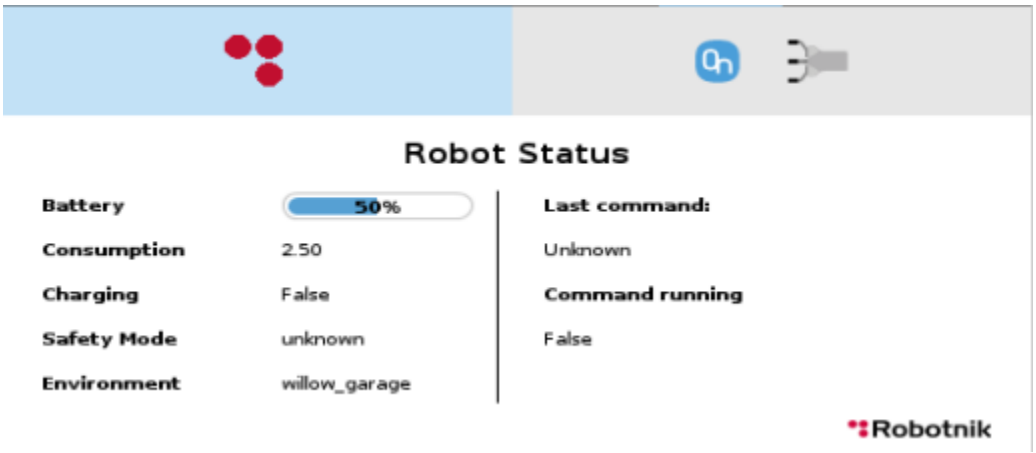


Figure 79. URCap toolbar

Robotnik

Developer manual

Revision v.1.4 - Noetic

Contents

1. Introduction

Welcome to the Developer manual. This document describes the main robot parameters, operations and development of robot software using ROS.

This document is dedicated to the advanced robot programmer, with a medium level of programming and familiarized with ROS.

Please, pay close attention to safety warnings before operating the robot.

For further information, you can consult the following documents:

- User manual of the robot
- Maintenance manual of the robot
- Control interface manual
- Appendices

⚠ This manual has been written using the RB-Kairos+ as example but it applies to the following robot models: RB-Kairos+, Summit XL, Summit XL Steel, RB-1 Base, RB-Theron, RB-Vogui.

1.1. Contact

If you have any doubt or problem with your robot, please contact us by sending an e-mail[1] to support@robotnik.es, indicating the serial number of your robot. We are constantly improving and upgrading our products and services. Any feedback from users is welcome.

NOTE: You can get in contact with us either in English or Spanish.

1.2. Robot Types

This manual is applicable to all the RB-Kairos+ series, except RB-Kairos+ 16e.
This manual is also applicable to Summit XL, Summit XL Steel, RB-1 Base, RB-Theron and RB-Vogui.

NOTE: This manual may include some images from different robots, as visual support of the applicable written information.

2. ROS architecture

ROS is an open-source meta-operating system for your robot that provides inter-process message passing services (IPC) in a network.

ROS is also an integrated framework for robots that provides:

- Hardware abstraction layer
- Low level device control
- Robot common functionality (simulation, vision, kinematics, navigation, etc.)
- IPC
- Package and stack management

ROS provides libraries and tools to ease the development of robot software in a multi-computer system.

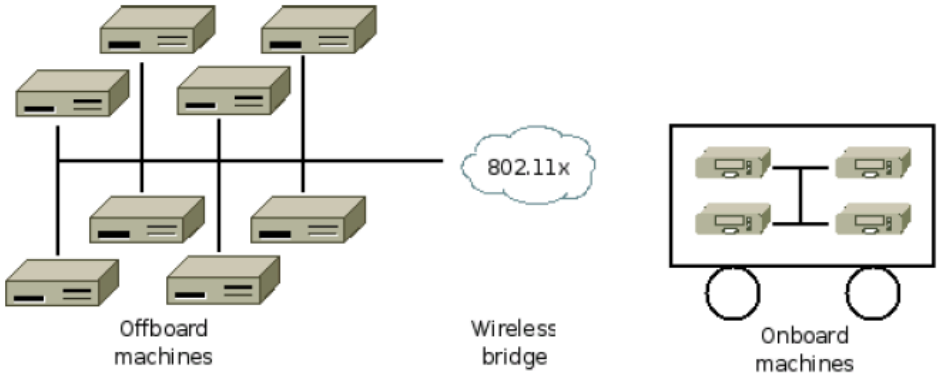


Figure 1. ROS Multi-computer schema

ROS offers a framework to solve common research and development needs of robot systems:

- Cooperation of different research groups
- Proven functionality
- Easy and robust access to robotics hardware

One of the main objectives of ROS is the code reusability. This objective is fulfilled by a large and growing community of developers that share their results worldwide, and by the inclusion of other robot frameworks (ROS integrates Player/Stage, Orocos, etc.) and other open-source components (like Gazebo or Openrave).

ROS integrates additional development tools like rviz (simulation of complete robots and environments with maps), rxgraph (visualization of node interconnection), rosbag (extremely useful data logging utility), etc.

For detailed systems descriptions, tutorials, and a really important number of stacks and packages, please visit www.ros.org.

You can find all the software developed by Robotnik Automation on ROS by visiting the official Github repository (https://github.com/RobotnikAutomation).

3. Robot packages

All the packages that the robot uses for its regular operation are located inside the workspace called catkin_ws, in the robot's HOME folder. The packages have been divided into different folders to organize them according to their functionality

- **command_manager**: A set of Robotnik packages intended to simplify interaction with the robot through simple string-based commands.
- **imu**: Packages containing IMU device drivers or associated tools.
- **manipulation**: Drivers and controllers for robotic arms and different types of tools and grippers.
- **navigation**: Packages to extend standard navigation functionality.
- **robot**: A set of packages that contains common configurations, URDFS, etc.
- **sensors**: Packages containing different sensor drivers or associated tools.
- **web**: Contains packages used by the HMI and web protocols such as REST or XMLRPC.
- **localization**: Packages to extend standard localization functionality.
- **msgs**: A set of packages that contain ROS msgs used by other packages.
- **others**: Here you will find all those packages that have not been classified in the other folders.
- **simulation**: Contains Gazebo packages, custom plugins and models for simulation.

In addition to the folders listed above, there are two additional folders:

- **robot_bringup**: It contains all the launch files and configurations (yaml) of the nodes that are launched at robot start-up. The autostart script (Autoboot sequence) references the files in this package.
- **debs**: It contains the proprietary software and all its dependencies in the form of ".deb". This software is installed on the robot by default.

4. Configuration

4.1. Advanced connection to the robot

You can connect to the robot via wi-fi network or ethernet connection using the LAN port in the rear panel. For further information, please consult the Connectivity chapter in the User manual.

This chapter explains several specific situations that need ROS knowledge.

4.1.1. CPU connection though ROS_MASTER_URI

You can get data and control the robot from a remote machine using ROS. This method has some requirements:

1. Install ROS (http://wiki.ros.org/noetic/Installation/Ubuntu)
2. Create a workspace and install packages from the robot workspace in your computer:
    a. All the packages contained in the msgs folder.
    b. rbkairos_description[1]
    c. summit_xl_description[2]
    d. robotnik_sensors

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
```

1. The description packages depend on the robot model (rb1_base_description, rb_theron_description, rbkairos_description, rbvogui_description).
2. RB-Kairos and Summit XL Steel also require summit_xl_description.

3. Download dependencies and compile the stacks:

```
cd ~/catkin_ws
rosdep install --from-path src --ignore-src -y -r
catkin build
```

4. Add the robot to your /etc/hosts file:

```
192.168.0.200 SXLSK-YYYYMMDDAA
```

5. Add your IP and hostname in the robot /etc/hosts file:

```
192.168.0.XX your-hostname
```

6. Run the following command to establish the ROS master in your machine

```
export ROS_MASTER_URI=http://192.168.0.200:11311
```

This configuration is valid if you are connected to the local network of the robot. If the robot is connected to another network and you are not using the local network, the IPs will be different.

1. Download dependencies and compile the stacks:

```
cd ~/catkin_ws
rosdep install --from-path src --ignore-src -y -r
catkin build
```

Once ROS is installed in your machine you will need to configure this machine to be able to connect to the robot ROS_MASTER.

1. Add the robot hostname (its serial number, normally lowercase letter) in your /etc/hosts file:

```
192.168.0.200   RobotHostname
```

2. Add your computer hostname in the robot /etc/hosts file:

```
YourIp   YourHostname
```

3. Start the robot, connect to its wifi network, open a terminal and type:

```
export ROS_MASTER_URI=http://SerialNumber:11311
cd ~/catkin_ws
source devel/setup.bash
```

4. If everything worked, you should be able to see all the topics published by the robot and the services offered by the robot controller:

```
rostopic list
rosservice list
```

5. You can view the values published by the nodes with rostopic echo, e.g.:

```
rostopic echo /tf
```

At this stage you can have a first look at the robot with

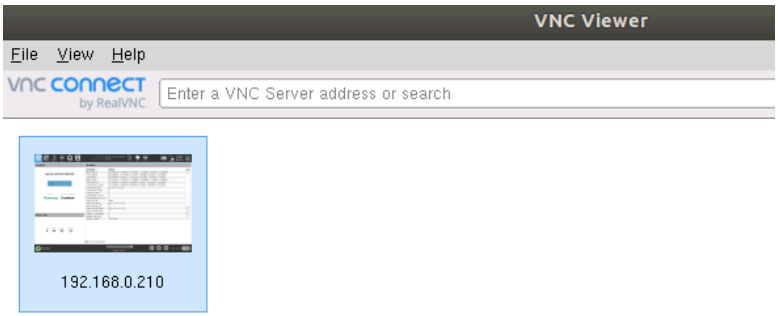
```
rqt
```

and

```
rosrun rviz rviz
```

4.1.2. UR arm connection via VNC

The UR arm is managed by Polyscope. It has installed a VNC server to enable remote access to the UR Polyscope. The VNC server starts automatically. The credentials and the IP address of the arm are detailed in the “Users and passwords” chapter.



In case of problems, if the VNC does not start, you can force the server from the base PC:

1. Make sure the arm is powered on.
2. Start the VNC server on the UR PC.

```
roscd robot_bringup/scripts && ./vnc_ur_arm.sh
```

Or

```
ssh root@192.168.0.210 bash -c './start_vnc.sh'
```

NOTE: The credentials for the PC of the arm are provided in the “Users and passwords” chapter.

3. Once the command, it will return the port where the VNC server is located.
4. Connect to the VNC server using the IP of the arm and the returned port.

4.2. Users and passwords

In this section you can find all the user and password configurations of the internal devices.

Mobile platform CPU	
User	robot
Password	R0b0tn1K

UR Arm	
User	root
Password	R0b0tn1K
Safety password	ur

Router[1]	
User	admin
Password	R0b0tn1K
SSID	SXLSK-YYYYMMDDAA[2]
Password	R0b0tn1K

1. This applies only if your robot has a Wi-Fi router.
2. The SSID will vary depending on the model and serial number of your product.

PTZ camera[1]	
User	root
Password	R0b0tn1K

1. This applies only if your robot has a PTZ camera.

4.3. Network

The robot has its own local network for all internal components. The following table shows an example with several devices connected to the local network of the robot and their IP addresses.

> ⚠ The network configuration will depend on the product you have purchased. For example, if your robot is not equipped with a router, there will be no visible Wi-Fi network to connect to.

Device	IP address
Mobile platform CPU	192.168.0.200
UR Arm	192.168.0.210
Router	192.168.0.1
PTZ camera front	192.168.0.185
PTZ camera rear	192.168.0.186
Front Laser	192.168.0.10
Rear Laser	192.168.0.11
Safety PLC	192.168.0.50

4.4. Autoboot sequence

The robot is ready to automatically launch all software when it is switched on. This is achieved through a system service called bringup-ros.

Some commands of interest to interact and monitor the status of the service are:

- Stop the service

```
sudo systemctl stop bringup-ros.service
```

- Start the service

```
sudo systemctl start bringup-ros.service
```

- Restart the service

```
sudo systemctl restart bringup-ros.service
```

- Get the status of the service

```
sudo systemctl status bringup-ros.service
```

The system service runs a script called bringup.sh in the robot's HOME folder. This script configures the robot and ROS environment and launches a series of launch files that are separated into different screens. Screen is a terminal multiplexer, which allows a user to access multiple separate terminal sessions inside a single terminal window or remote terminal session (such as when using SSH).

Some commands of interest to interact with the different screens:

- List all available screens

```
screen -list
```

- Attach a screen

```
screen -r <name_of_the_screen>
```

- Detach a screen (requires being in an attached screen)

```
CTRL+A+D
```

- Restart a screen (requires being in an attached screen)
    - Stop the screen

```
CTRL+C
```

    - Launch again

```
r
```

More specific information on the use and configuration of the screen command can be found at https://help.ubuntu.com/community/Screen.

4.5. Environment variables

The boot script mentioned in the previous section reads a list of environment variables that are used to configure the software start-up. These environment variables are divided into different files in the robot_params folder that is located in the robot HOME folder.

Environment variables are configured and customized during the robot installation process. Please note that modifying them may affect the normal operation of the robot. If you want to modify any variable, make sure that the modifications are correct.

The different robot models share the same environment variables (with different values to fit its characteristics). The values of the environment variables listed in the following subsections are the **default values for a RB-KAIROS+**. Your robot may have some different values set up during the installation process and depending on the configuration of your specific robot.

If you have any doubts, please do not hesitate to contact support@robotnik.es.

4.5.1. Bringup

The bringup.env file includes the environment variables that configure which screens are launched and the ROS package to be used as bringup.

Parameter	Description	Default value
ROBOT_RUN_SIMULATION	Run simulation or real robot. Always true for real robot	false
ROBOT_RUN_ROBOT_LOCAL_CONTROL	Flag to start Robot Local Control node	true
ROBOT_RUN_LOCALIZATION	Flag to start Localization nodes	true
ROBOT_RUN_PERCEPTION	Flag to start Perception nodes	true
ROBOT_RUN_NAVIGATION	Flag to start Navigation nodes	true
ROBOT_RUN_ROSTFUL_SERVER	Flag to start Rostful nodes	true

ROBOT_RUN_HMI	Flag to start HMI nodes	true
ROBOT_RUN_COMMAND_MANAGER	Flag to start Command Manager nodes	true
ROBOT_RUN_SENSORS	Flag to start Sensors nodes	true
ROBOT_RUN_ARM	Flag to start Manipulation nodes	true
ROBOT_WORKSPACE	Path to the ROS workspace that the robot is using	/home/robot/catkin_ws
ROBOT_BRINGUP_PACKAGE	Name of the ROS package where all launch and yaml files are located	robot_bringup
ROBOT_BRINGUP_SIM_PACKAGE	Name of the ROS package where all simulation launch files are located. Not used for the real robot.	summit_xl_sim
ROBOT_BRINGUP_LAUNCH	Name of the ROS launch used as bringup of the base software of the robot.	robot_complete.launch
ROBOT_BRINGUP_SIM_LAUNCH	Name of the ROS launch used as bringup of the simulation of the robot	rbkairos_complete.launch

4.5.2. Robot

The robot_params.env contains the environment variables related to the screen called bringup. This screen launches the basic robot software: robot_state_publisher, hardware_interface (robotnik_base_hw), robot controller, battery_estimation, etc.

To access to the bringup screen:

```
screen -r bringup
```

Parameter	Description	Default value
ROBOT_ID	Robot prefix and namespace to be used.	robot
ROBOT_MODEL	Name of the robot model.	rbkairos
ROBOT_XACRO	Name of the URDF file used to load the robot description. This file is located in the rbkairos_description package. The URDF filename may differ depending on the real robot.	rbkairos_base.urdf.xacro
ROBOT_KINEMATICS	Defines the kinematics of the robot. If the robot has rubber wheels, the kinematics will be **diff**. If the robot has mecanum wheels, the kinematics will be **omni**.	omni
ROBOT_GEARBOX	Indicates the motor gearbox.	9.56
ROBOT_WHEEL_DIAMETER	Indicates the wheel diameter depending on the type of wheels. Rubber (0.22) or mecanum wheels (0.25).	0.25
ROBOT_TRACK_WIDTH	Distance between wheels on the same axis.	0.538
ROBOT_WHEEL_BASE	Distance between front and rear wheels.	0.43
ROBOT_HAS_ELEVATOR	Flag to set the robot has an elevator (normally to pick & place carts).	false
ROBOT_HAS_ENCODER	Flag to set whether or not the	true

	motors have encoders.	
ROBOT_HAS_SAFETY_MODULE	Flag to set whether the robot has safety lasers and/or safety plc.	true
ROBOT_SAFETY_LASER_MODEL	Different safety configurations based on laser safety laser.	sick_microscan3
ROBOT_BASE_HW_BATTERY_VOLTAGE_OFFSET	The real battery voltage (multimeter) minus the one returned by the drive. Deprecated, not used.	0.0
ROBOT_K_ANALOG_INPUTS_MULTIPLIERS	K multipliers for analog inputs of the drivers. The first input refers to battery voltage input. The second input refers to current consumption. The values may differ for each robot.	[16.6,-12.5,1.0,1.0]
ROBOT_PAD_MODEL	Indicates which model of joystick the robot is using.	ps4
ROBOT_PAD_DEADZONE		0.12
ROBOT_PAD_DEV	Input from which the joystick data is read.	/dev/input/js_base
ROBOT_HAS_LEDS	Flag to set if the robot has leds or not.	true
ROBOT_LEDS_PORT	Serial port where the leds are connected.	/dev/ttyUSB_LEDS

The bringup screen also uses battery_params.env file:

Parameter	Description	Default value
ROBOT_BMS_MODEL	Indicates the model of the BMS that the robot is using. The model equals to the name of the launch file in the **$ROBOT_BRINGUP_PACKAGE**	none
ROBOT_BMS_PORT	Port of the BMS if it is connected using USB.	/dev/ttyUSB_BMS
ROBOT_BATTERY_VOLTAGE	Voltage of the battery.	48
ROBOT_READ_VOLTAGE_FROM_ANALOG_INPUT	Flag to read voltage from analog input of the drivers.	true
ROBOT_VOLTAGE_ANALOG_INPUT_NUMBER	Analog input of the driver to read the voltage.	1
ROBOT_CURRENT_ANALOG_INPUT_NUMBER	Analog input of the driver to read the current.	2
ROBOT_DOCKER_MODE		automatic_sw
ROBOT_BATTERY_INVERTED_CONTACT_RELAY	Flag to invert the value of input of the contact relay.	false
ROBOT_BATTERY_CONTACT_RELAY_INPUT_NUMBER	Digital input number to detect the contact between the robot and the docking station charger.	2
ROBOT_BATTERY_CHARGER_RELAY_OUTPUT_NUMBER	Digital output number to switch on the internal robot charging contacts.	1

4.5.3. Sensors

The sensors_params.env contains the environment variables related to the screen called sensors. This screen launches all ROS nodes that manage the communication with the sensors: GPS, lasers, IMU, camera, etc.

The robot may be equipped with several sensors of the same type, in which case environment variables with the same name identified by a number are used. In this manual, only the variables for the first sensor of each type are listed.

To access to the sensors screen:

```
screen -r sensors
```

Parameter	Description	Default value
ROBOT_LASER_1_MODEL	Indicates the model of the laser 1 that the robot is using. The model equals to the name of the launch file in the **$ROBOT_BRINGUP_PACKAGE**	hokuyo_ust
ROBOT_LASER_1_IP	IP address of the laser 1 if it is connected using Ethernet.	192.168.0.10
ROBOT_LASER_1_PORT	Port of the laser 1 if it is connected using USB.	-
ROBOT_LASER_1_ID	Namespace and prefix for the laser 1.	front_laser
ROBOT_LASER_1_MIN_ANGLE	Minimum angle of the laser 1.	-
ROBOT_LASER_1_MAX_ANGLE	Maximum angle of the laser 1.	-
ROBOT_CAMERA_1_MODEL	Indicates the model of the camera 1 that the robot is using. The model equals to the name of the launch file in the **$ROBOT_BRINGUP_PACKAGE**	orbbec
ROBOT_CAMERA_1_IP	IP address of the camera 1 if it is connected using Ethernet.	192.168.0.185

ROBOT_CAMERA_1_CALIBRATION	Name of the calibration file used for the camera 1.	astra_s
ROBOT_CAMERA_1_ID	Namespace and prefix for the camera 1.	front_rgbd_camera
ROBOT_CAMERA_1_DEVICE_ID	Device ID for camera 1.	#1
ROBOT_IMU_MODEL	Indicates the model of the IMU that the robot is using. The model equals to the name of the launch file in the **$ROBOT_BRINGUP_PACKAGE**.	pixhawk
ROBOT_IMU_PORT	Serial port of the IMU.	/dev/ttyUSB_IMU
ROBOT_RUN_IMU_COMPLEMENTARY_FILTER	Flag to run IMU complementary filter.	true
ROBOT_GPS_MODEL	Indicates the model of the GPS that the robot is using. The model equals to the name of the launch file in the **$ROBOT_BRINGUP_PACKAGE**.	none
ROBOT_GPS_CONFIG	Name of the file for the GPS configuration.	none

4.5.4. Navigation

The navigation_params.env contains the environment variables related to the screen called navigation. This screen launches all ROS nodes related to the navigation and movements of the robot: nodes developed by Robotnik, move_base, laser filters for navigation, etc.

To access to the navigation screen:

```
screen -r navigation
```

Parameter	Description	Default value
ROBOT_HAS_DOCKER	Flag to run robotnik_docker	true
ROBOT_RUN_MOVE	Flag to run robotnik_move	true
ROBOT_RUN_MOVE_BASE	Flag to run move_base	true
ROBOT_MOVE_BASE_LOCAL_PLANNER	Name of the local planner used by move_base	teb
ROBOT_NAVIGATION_2D_SCAN_1	Topic of the first laser filtered used for navigation.	**$ROBOT_LASER_1_ID**/scan_filtered
ROBOT_NAVIGATION_2D_SCAN_2	Topic of the second laser filtered used for navigation.	none
ROBOT_NAVIGATION_2D_SCAN_2	Topic of the third laser filtered used for navigation.	none
ROBOT_RUN_LASER_FILTERS	Flag to run navigation laser filters	true

4.5.5. Localization

The localization_params.env contains the environment variables related to the screen called localization. This screen launches all ROS nodes related to the localization of the robot: amcl, multimap_server, slam_gmapping, etc.

To access to the localization screen:

```
screen -r localization
```

Parameter	Description	Default value
ROBOT_LOCALIZATION_SCAN_TOPIC	The scan topic that amcl is using to locate the robot.	**$ROBOT_LASER_1_ID**/scan
ROBOT_LOCALIZATION_ODOM_MODEL	Type of odometry used for the localization algorithm (mainly amcl)	**$ROBOT_KINEMATICS**

4.5.6. Perception

The perception_params.env contains the environment variables related to the screen called perception. This screen launches all ROS nodes related to the perception: Robotnik locators and scan filters.

To access to the perception screen:

```
screen -r perception
```

Parameter	Description	Default value
ROBOT_RUN_AR_LOCATOR	Flag to run alvar locator node.	true
ROBOT_RUN_REFLECTOR_LOCATOR	Flag to run laser locator node.	true
ROBOT_LASER_MODEL_INTENSITY_FILTER	Laser model used to filter intensity readings to detect reflectors.	**$ROBOT_LASER_1_MODEL**
ROBOT_DOCKING_STATION_TAG_DISTANCE	Distance between reflectors to identify a pair of reflectors as a docking station	0.3
ROBOT_DOCKING_STATION_TAG_MAX_DISTANCE_DETECTION	Maximum distance to detect docking stations.	3.5
ROBOT_PERCEPTION_SCAN_TOPIC	Scan topic to detect reflectors.	**$ROBOT_LASER_1_ID**/scan
ROBOT_PERCEPTION_CAMERA_TOPIC	Image topic to detect QR codes.	**$ROBOT_CAMERA_1_ID**/rgb/image_raw
ROBOT_PERCEPTION_CAMERA_INFO_TOPIC	Image info topic to detect QR codes.	**$ROBOT_CAMERA_1_ID**/rgb/camera_info

4.5.7. Robot Local Control (RLC)

The rlc_params.env contains the environment variables related to the screen called rlc. This screen launches the Robot Local Control node.

To access to the Robot Local Control screen:

```
screen -r rlc
```

Parameter	Description	Default value
ROBOT_RLC_ROSTFUL_SERVER_IP	IP address of the ROStful server.	127.0.0.1
ROBOT_RLC_ROSTFUL_SERVER_PORT	Port of the ROStful server.	8080
ROBOT_RLC_CHARGE_DOCK_OFFSET_X	Distance offset for the dock action when going to charge using ChargeComponent	-0.52
ROBOT_RLC_CHARGE_DOCKER_NAMESPACE	Docker namespace to perform the charge using ChargeComponent	omni_docker
ROBOT_RLC_PICK_DOCK_OFFSET_X	Distance offset for the pick action.	-0.3
ROBOT_RLC_PICK_STEP_IN_DISTANCE	Distance to move in after the dock in the pick action.	0.62
ROBOT_RLC_PLACE_STEP_OUT_DISTANCE	Distance to move out in the place action	-0.7
ROBOT_RLC_LOC_INIT_ENV	Initial environment loaded by the localization component	willow_garage

5. Advanced functionality

5.1. Mapping

The robot is able to map its environment thanks to the lasers it is equipped with.

First, ensure that the HMI is not executing any localization process.

```
ROS_NAMESPACE=robot roslaunch robot_bringup slam_gmapping.launch
```

Once you have the desired map, execute the following command to save it:

```
ROS_NAMESPACE=robot roslaunch robot_bringup map_saver.launch
```

It will save your in the user home directory as map.yaml and map.png, but you can custom the destination folder and the name of the map as following:

```
ROS_NAMESPACE=robot roslaunch robot_bringup map_saver.launch
map_name:=my_map destination_folder:=/path/to/map/folder
```

WARNING: The map is empty

- Make sure the rest of the robot software is running. Try moving it with the remote control.
- Check if the laser is publishing data. (*rostopic echo /robot/front_laser/scan*)
- Make sure that slam_gmapping node is subscribing to the right scan topic. (*rostopic info /robot/front_laser/scan*)

RB-Kairos+, Summit XL and Summit XL Steel use ***robot_bringup*** package for mapping.

5.2. Localization

First of all, you will need to check that the HMI is not running any localization process.

Then, load the map you are going to work with. To do this, execute:

```
ROS_NAMESPACE=robot roslaunch robot_bringup map_server.launch
prefix:=robot_
```

The available maps can be found in the *robot_bringup* package, in the *maps* folder. If you want to change the map that is loaded:

```
ROS_NAMESPACE=robot roslaunch robot_bringup map_server.launch
prefix:=robot_ maps_path:=path_to_map map_file:=file_to_load.yaml
```

If you want to work with a map that has been created using the HMI, they are saved in the maps folder of robot_bringup package by default.

WARNING: The map is not loaded

If the path or the file name is not correct, the terminal will print an error saying “Map server could not open /the/path/that/is/using/to/load/the/map“

- Make sure that the map filename is correct.
- Make sure that the maps_path is correct
- If you have changed the name of the map file, you will need to edit the map_file.yaml too. This yaml file contains a reference to the pgm file.
- The map frame used is *robot_map*.

Once you have the map loaded, you need to run the localization program:

```
ROS_NAMESPACE=robot roslaunch robot_bringup amcl.launch
```

If the amcl is capable of reading the map, the node will print that it received the map after requesting it.

WARNING: The localization is not working

- Check that the map_server is running. (*rosnode list | grep map_server*)
- Check that the map is being published (*rostopic echo /robot/map*)
- Check if the laser is publishing data in the following topics (*rostopic echo /robot/front_laser/scan rear_laser/scan | merged_laser/scan , top_3d_laser/scan*)
- Make sure that the localization node is subscribing to the right scan topic. (*rostopic info /robot/fornt_laser/scan*)

RB-Kairos+, Summit XL and Summit XL Steel use ***robot_bringup*** package for localization.

5.3. Navigation

The robot can perform two types of autonomous movements depending on whether they are map based (move_base package) or not map based (robotnik_navigation package).

5.3.1. Robotnik_navigation package

⚠ This type of movement does not avoid obstacles. The robot will stop for safety if it is equipped with the hardware safety package and an obstacle within the safety range.

5.3.1.1. Move

This component is intended to perform basic movements. These movements are based on the odometry frame of the robot. They allow the robot to rotate on itself, to move forward/backward and sideways (if it has omni wheels).

⚠ Try to send only linear (x and/or y) or angular displacement. The combination of linear and angular displacement is not allowed.

Allows the robot to perform linear and angular movements based on its odometry. Messages related to the *Move* action are defined in the *Move.action* file of the *robotnik_navigation_msgs* package:

The configurable parameters are located in the *config/navigation/move* folder in the *robot_bringup* package. On robotnik_move section you can find a list with all available node parameters with their description and default values. This action uses a custom ROS Message whose structure is shown on Move.action section.

The action can be run via the actionlib client:

```
rosrun actionlib_tools axclient.py /robot/move
```

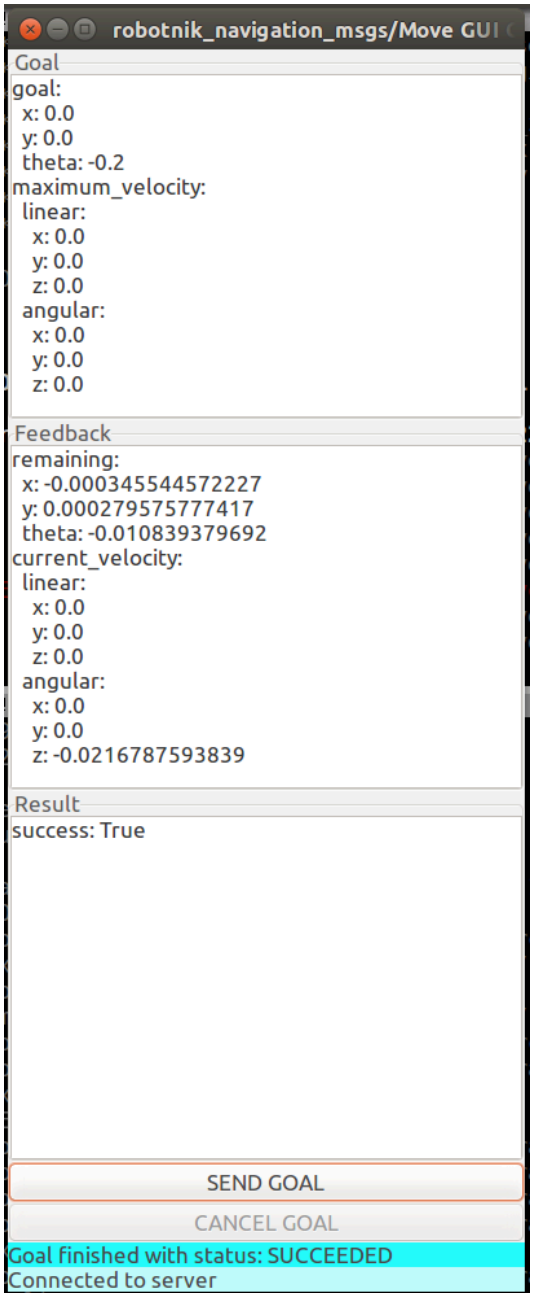


Figure 2. Move axclient

5.3.1.2. Dock

This component is intended to perform basic movements to reach a target. It allows the robot to approach an identified *frame/target*, normally for docking purposes. Messages related to the *Dock* action are defined in the *Dock.action* file of the *robotnik_navigation_msgs* package:

The configurable parameters are located in the *config/navigation/dockers* folder in the *robot_bringup* package. On the robotnik_dock section you can find a list with all available node parameters with their description and default values. This action uses a custom ROS Message whose structure is shown on the Dock.action section.

There are different instances of the docker node:

- pp_docker: PurePursuite docker. Performs the docking using linear (X) and angular commands.
- omni_docker: Omni-directional docker. This docker is only run if your robot has mecanum wheels. Performs the docking using linear (X and Y) and angular commands

⚠ Diff_docker is no longer in use and its use is not recommended. Use pp_docker instead for a better experience.

In order to launch a docking action to any kind of marker or an arbitrary frame open a terminal and execute (where *docker* can be substituted by any of the instances):

```
rosrun actionlib_tools axclient.py /robot/docker
```

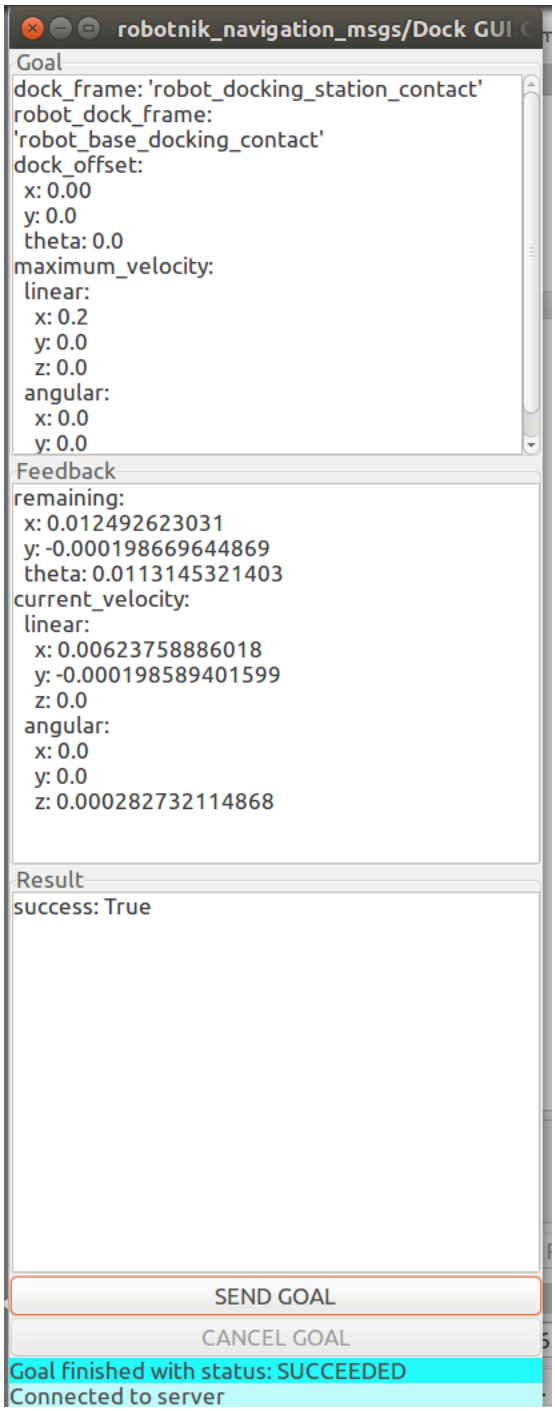


Figure 3. Docker axclient

5.3.2. move_base package

The move_base package provides an implementation of an action that, given a goal in the map, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. The local planner used is the TEB that will be detailed later.

The configurable parameters are located in several files in the *config/navigation* folder of the *robot_bringup* package. The main file related to move_base is named *move_base_params.yaml* .

WARNING: The navigation is not working

- Check that the map_server node is running. (*rosnode list | grep map_server*)
- Check that the localization node is running. (rosnode list | grep amcl)

5.3.2.1. TEB

The initial trajectory generated by a global planner is optimized during runtime w.r.t. minimizing the trajectory execution time (time-optimal objective), separation from obstacles and compliance with kinodynamic constraints such as satisfying maximum velocities and accelerations.
The configurable parameters are located in several files in the *config/navigation* folder of the *robot_bringup* package depending on the wheel configuration (diff or omni):

- Maximum velocities and accelerations
- Minimum obstacle distance
- Goal tolerances

5.4. Arm manipulation

This chapter only applies to RB-Kairos+ and RB-Vogui series equipped with an Universal Robots arm.

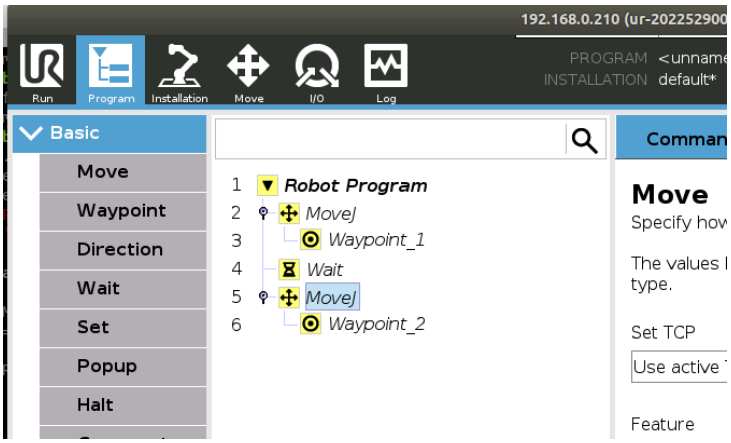
The Universal Robot arms can be controlled in two ways depending where the control is executed:

- From UR Polyscope using predefined commands and urscript.
- From the robot CPU using the UR ROS official driver and the external control URCap.

The safety and safeguard signals provided by the robot, such as emergency stop, safeguard stop, etc (these signals depend on the type of robot), are independent of this control.

5.4.1. UR Polyscope

The UR Polyscope allows the developer to create a program using the urscript.



The main advantage of this method is that you can use all the Universal Robot functions and URCaps. The trajectories and control is executed by the arm, providing the best arm performance.

On the other hand, the main drawback is that the arm is not aware of the environment and the collisions in the trajectory calculation. There is not a direct control integration between the arm and the robot base.

For further information, please consult the UR official manuals.

5.4.2. UR ROS driver

The ROS control option is the default mode in Robotnik configuration.

The main benefits of this mode are:
- The UR ROS driver allows the robot to be aware of the environment, specifically for collision avoidance.
- It is completely integrated with ROS and the robot base.

Nevertheless, this option has several drawbacks:
- The performance of trajectories and speeds are not optimized.
- The latency of the communication between the base and the arm may affect the movement performance.

The UR joints and speed limits of the UR installation will always take preference over the ROS driver calculations.

⚠ Pay attention to arm trajectories and configuration before sending a new trajectory.

The robot launches the UR ROS Driver in order to establish communication with the arm. From the arm side, a script called *external_control* is launched to receive commands from the robot. When the arm is connected to the robot using ROS, the trajectories can be sent using the following action:

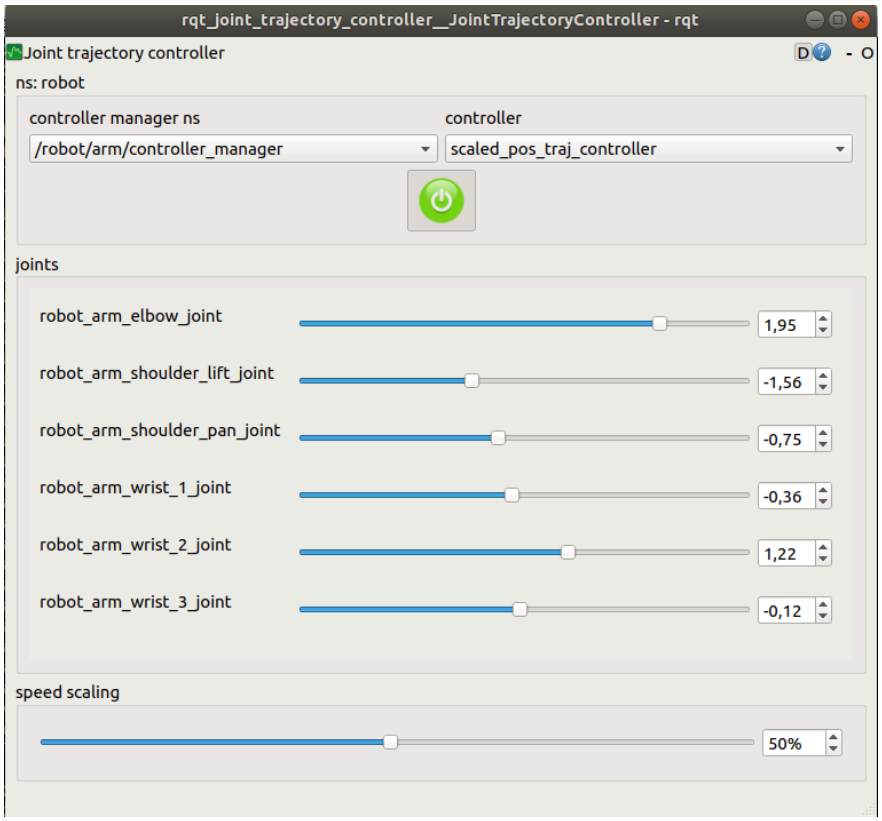
```
/robot/arm/scaled_pos_traj_controller/follow_joint_trajectory/cancel
/robot/arm/scaled_pos_traj_controller/follow_joint_trajectory/feedback
/robot/arm/scaled_pos_traj_controller/follow_joint_trajectory/goal
/robot/arm/scaled_pos_traj_controller/follow_joint_trajectory/result
/robot/arm/scaled_pos_traj_controller/follow_joint_trajectory/status
```

5.4.2.1. Normal operation for ROS arm control

When the robot base and arm are powered on, all the programs for the manipulation are started automatically.

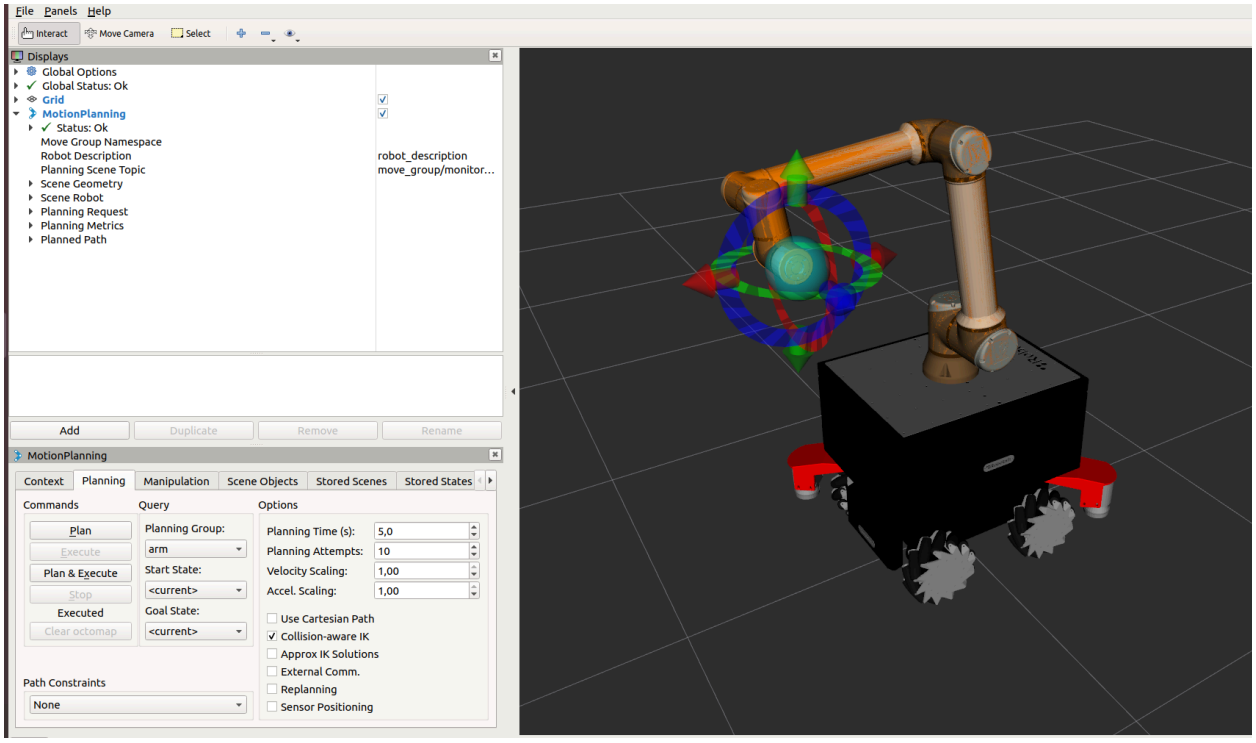
Use *rqt_joint_trajectory_controller* in order to move the arm joint by joint:

```
ROS_NAMESPACE=robot rosrun rqt_joint_trajectory_controller
rqt_joint_trajectory_controller
```



Or use Moveit to send trajectories to the arm. The name of the package depends on the name of the robot.

```
ROS_NAMESPACE=robot roslaunch <robot model>_ur_moveit demo.launch
```



5.4.2.2. Polyscope setup for ROS arm control

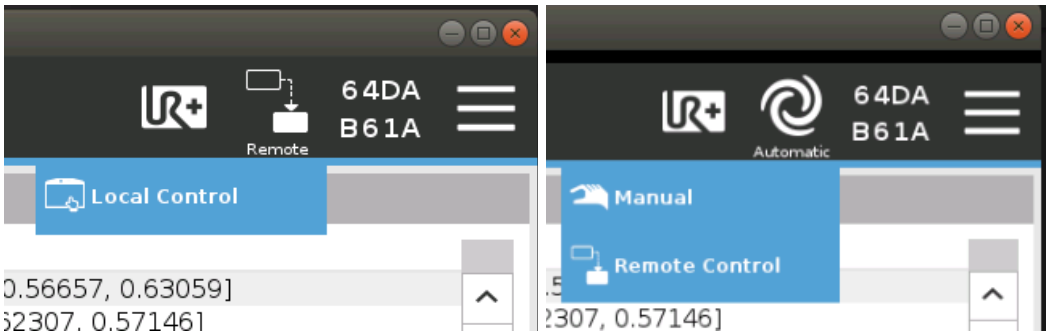
The polyscope setup is already performed in all Robotnik robots on the Robotnik default installation. However, in some cases, it could be required to perform this setup again.

⚠ It is recommended to perform a backup of polyscope configuration before making any change in the robotic arm. A copy of programs and installations can be useful too. Please consult UR documentation for further information.

⚠ The modification of Robotnik software installation parameters could invalidate the safety and risk assessment, and could affect the guarantee of the robot. Please pay attention when modifying installation parameters.

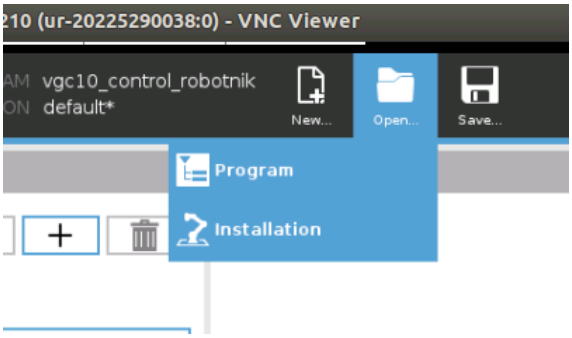
---

1. Change the control mode to Local Control and then to Manual.

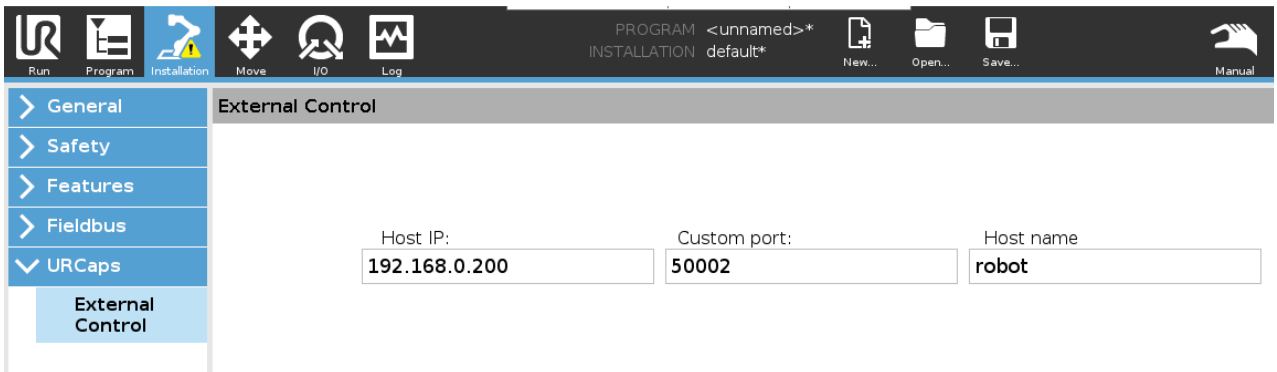


The manual mode will ask for a password. Please check the “Users and passwords” chapter on this manual for further information.

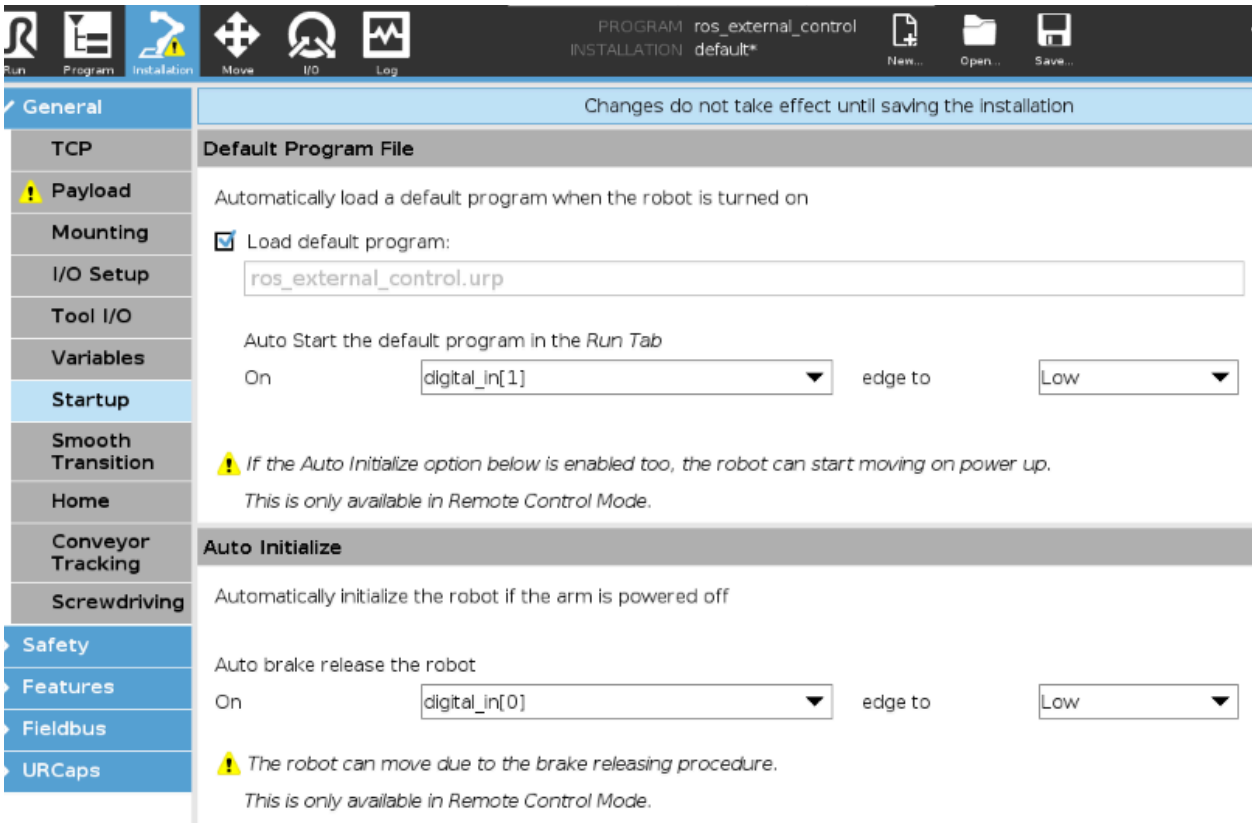
2. Load the *ros_external_control.urp* program:



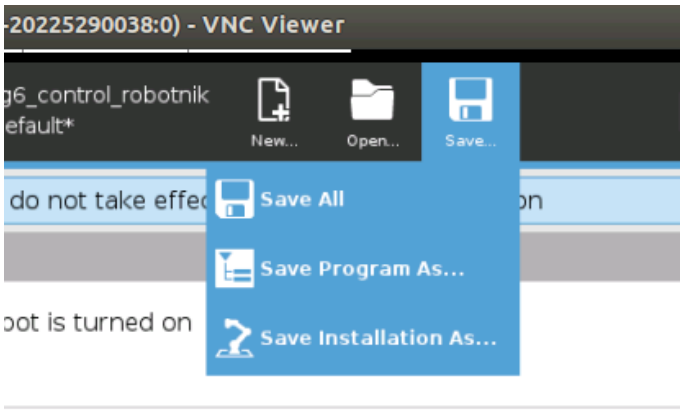
3. Check the external control parameters: Go to *Installation → URCaps → External Control*. The Host IP must be the CPU address (192.168.0.200) and the custom port 50002.



4. Go to *Installation* → *General* → *Startup* and set the *ros_external_control.urp* program. With this configuration, *ros_external_control.urp* will be launched automatically when the control box boots.

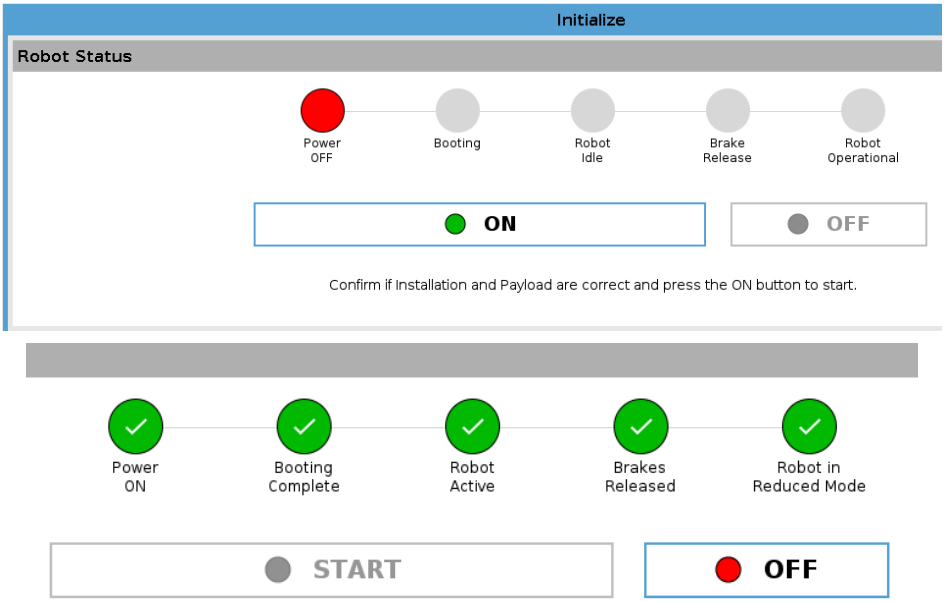


5. Save the installation in Save → Save Installation As. Then overwrite the default installation:

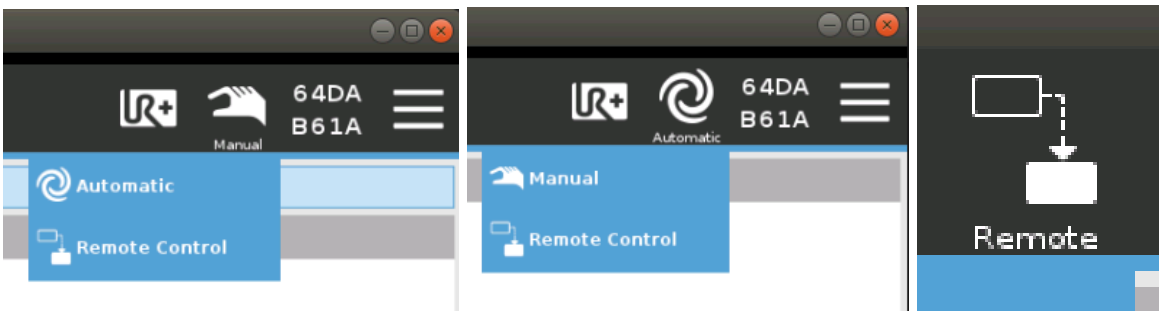


⚠ Do not change the installation name. “default.installation” is the installation that will be charged automatically when booting the UR arm.

6. Turn on the arm in the Polyscope. Click on the Power off button on the lower left corner of the screen. Click on the ON button and then on the START button. The brakes will be released and the arm will be ready to move.



7. In the Polyscope, change the control mode to Automatic and then to Remote control:



5.4.2.3. Robot setup for ROS arm control

1. In the robot CPU, run the bringup script in order to launch all the packages.

```
cd /home/robot
./bringup.sh
```

2. When the script is launched completely, test the arm moving it joint by joint.

```
ROS_NAMESPACE=robot rosrun rqt_joint_trajectory_controller
rqt_joint_trajectory_controller
```

3. If the joint by joint control worked, test the arm using MoveIt:

   The standard robots have a pre-installed MoveIt package ready to send trajectories to the arm. The name of the package depends on the name of the robot.

```
ROS_NAMESPACE=robot roslaunch <robot model>_ur_moveit demo.launch
```

   For example, for a RB-Kairos+ with a UR10e arm:

```
ROS_NAMESPACE=robot roslaunch rbkairos_ur_moveit demo.launch
```

   Or for a RB-Vogui with a UR10e arm:

```
ROS_NAMESPACE=robot roslaunch rbvogui_xl_ur_moveit demo.launch
```

5.4.2.4. Troubleshooting for ROS arm control

Sometimes the programs in the robot base or in the robot arm are not started automatically for different reasons.

- UR Polyscope program not launched automatically

Connect to the Polyscope using the following service from the ROS side:

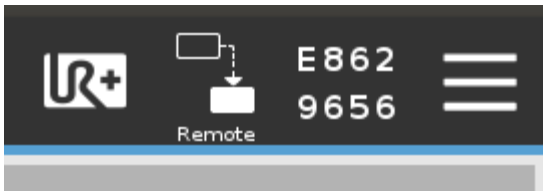
```
rosservice call /robot/arm/ur_hardware_interface/dashboard/connect "{}"
```

Then, launch the program loaded in the Polyscope:

```
rosservice call /robot/arm/ur_hardware_interface/dashboard/play "{}"
```

- The arm does not boot automatically or the UR Polyscope rejects the ROS services commands:

Make sure that the Polyscope is in remote control mode:



- Port already in use from the robot base side:

The port is already in use due to an unexpected port disconnection. The error can be visualized in the arm_bringup screen.

```
screen -r arm_bringup
```

Launch again the bringup.sh script

```
./bringup.sh
```

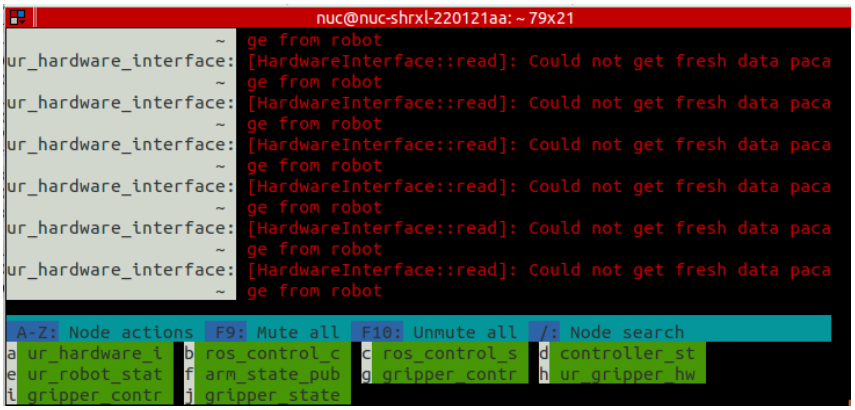
If the port does not close, restart the robot CPU.

```
sudo reboot
```

- Could not get port fresh data

The error can be visualized in the arm_bringup screen.

```
screen -r arm_bringup
```



Launch again the bringup.sh script

```
./bringup.sh
```

5.5. Gripper manipulation

This chapter only applies to RB-Kairos+ and RB-Vogui series equipped with an Universal Robots arm.

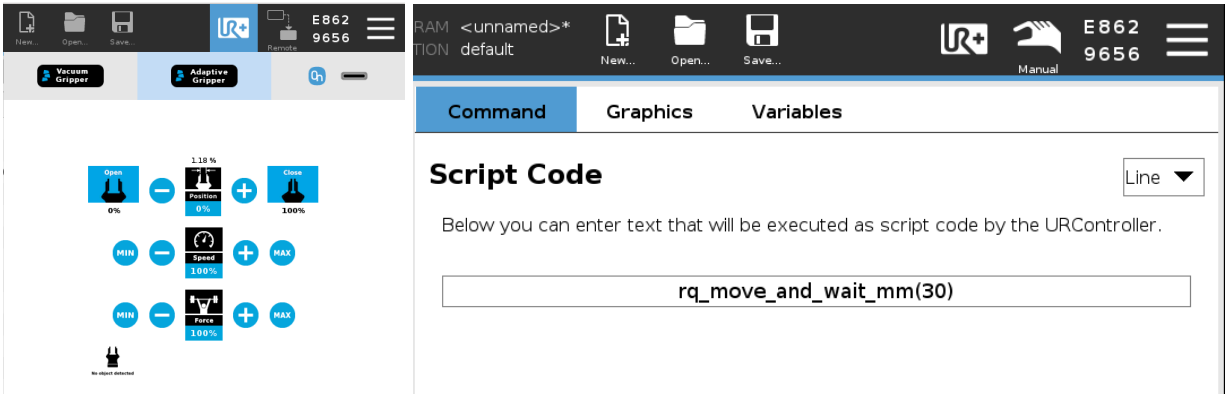
The Universal Robot arm can control different grippers. There are two possible ways to control them:

- From UR Polyscope using an urscript and the gripper URCap.
- From ROS using a xmlrpc server and the gripper URCap. This option is not available by default.

⚠ The gripper ROS control is not implemented in the standard Robotnik software installation. It is only available for specific models and upon request. Please, contact support@robotnik.es for further information.

5.5.1. UR gripper Polyscope

Some grippers can be controlled from UR Polyscope, through their own urcap. They can be controlled using the interface and default commands, or using an urscript with their specific functions.

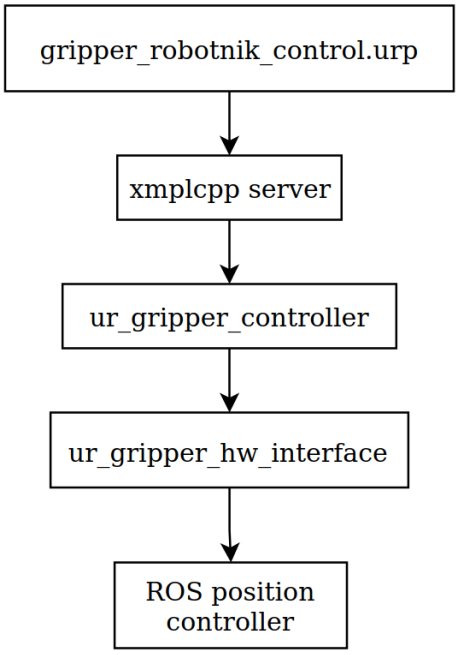


For further information, please consult the gripper and UR official manuals.

5.5.2. ROS gripper control

This chapter is only applicable to grippers and arm combinations with ROS control and upon request.

Some grippers can be controlled from UR Polyscope through a xmlrpc server using a ROS controller. The gripper communication between ROS and Polyscope follows the main structure shown below:



- **gripper_robotnik_control**: this script must be run in the Polyscope program in order to establish connection to the ROS using an xmplcpp server. It is composed by the external control program to control the arm from the ROS side and the gripper code to control it using the xmplcpp server.

- **xmplcpp server**: this server is the interface between Polyscope and ROS. The server is created from the ROS side by the ur_gripper_controller package

- **ur_gripper_controller**: this package uses a xmplcpp server to communicate with the Polyscope and creates a basic ROS interface in order to send rostopic commands.

- **ur_gripper_hw_interface**: it uses the rostopic commands created by the ur_gripper_controller in order to communicate with the robot hardware interface library.

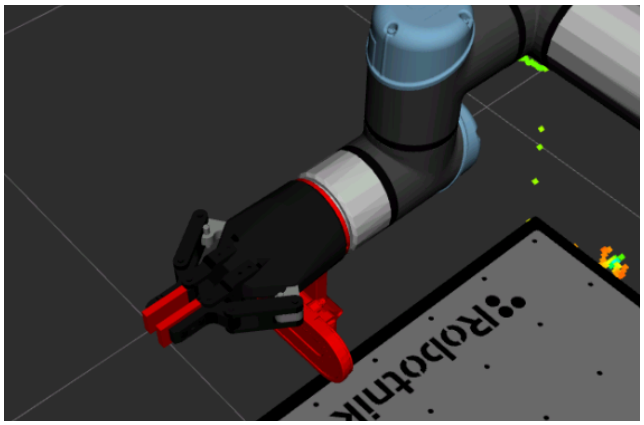
- **ROS position controller**: the robot hardware interface creates a rostopic position controller in order to send commands to the gripper.

5.5.2.1. Usual operation for ROS gripper control

With Robotnik installation, the needed manipulation programs are started automatically after robot base and arm power-on.

Move the gripper using the ROS position controller. Depending on the type of the joint, the units may be in **meters** or **radians**.

```
rostopic pub /robot/gripper/command std_msgs/Float64 "data: 0.0"
```



5.5.2.2. Polyscope setup for ROS gripper control

Check that the specific program for the gripper is loaded in the Polyscope, and load it if needed. For further information, you can follow the Polyscope setup for ROS arm control chapter.
Load the *gripperX_control_robotnik.urp* corresponding to the desired gripper:

> ⚠ The program will only be available if the gripper ROS integration is included in the standard Robotnik software installation, upon request.

5.5.2.3. Robot setup for ROS gripper control

1. In the robot CPU, run the bringup script in order to launch all the packages.

```
cd /home/robot
./bringup.sh
```

2. Move the gripper using the ROS position controller.

```
rostopic pub /robot/gripper/command std_msgs/Float64 "data: 0.0"
```

5.5.2.4. Troubleshooting for ROS gripper control

As the gripper depends on the arm, please consult the *Troubleshooting for ROS arm control* chapter.

5.6. Robot Local Control

Robotnik robot local control is a software that manages the overall state and behavior of the robot. This is done by monitoring the health status of several ROS components, and proceeding according to this status and the actions the robot is required to perform.

It also provides a set of components that allow the execution of procedures. A procedure is identified by an id, which is set by the robot_local_control and returned to the one who added the procedure. After a procedure is added, its state can be queried using its id (-1).

5.6.1. Configuration

The package is already configured with the correct parameters and they can be found in the config/robot_local_control folder in *robot_bringup* package. However, there are some parameters that you can tune according to your needs and they can be found in the '*navigation.yaml*' file. These parameters and their meaning is explained in the following sections.

5.6.2. Procedures

A procedure is an action that is performed by the robot. This action can be composed of one or multiple different actions.
These procedures have common services:

- **cancel**: It cancels the current procedure
- **query_state**: It asks the state of the current procedure

```
rosservice call
/robot/robot_local_control/NavigationComponent/Component/cancel "header:
id: -1
priority: 0
stamp:
  secs: 0
  nsecs: 0
name: ''"
```

5.6.3. GOTO

This component connects to an instance of ROS move_base to send navigation goals to it.

On 7.1.5. Robot Local Control chapter on GoTo section you can find a list with all available node parameters with their description and default values.

5.6.3.0.1. Add request

In order to initiate the GoTo procedure you have to call the following service:

```
rosservice call
/robot/robot_local_control/NavigationComponent/GoToComponent/add
"procedure:
  goals:
  - header:
      seq: 0
      stamp:
      secs: 0
      nsecs: 0
```

```
      frame_id: 'robot_map'
      pose:
        x: 0.0
        y: 0.0
        theta: 0.0
  max_velocities:
  - linear_x: 0.0
    linear_y: 0.0
    angular_z: 0.0"
```

The frame_id parameter specifies the map frame where the robot is going to navigate. Then, the pose within the map that the robot should reach and the maximum velocity allowed to perform this action (0.0 velocity means that it uses the maximum velocity established in move_base configuration).

5.6.3.1. Move

This component connects to an instance of move to send goals to it.

On 7.1.4. Robot Local Control chapter on Move section you can find a list with all available node parameters with their description and default values.

5.6.3.1.1. Add request

In order to initiate the move procedure you have to call the following service:

```
rosservice call
/robot/robot_local_control/NavigationComponent/MoveComponent/add
"procedure:
  goal:
      x: 0.5
      y: 0.0
      theta: 0.0
  maximum_velocity:
      linear:
        x: 0.0
        y: 0.0
        z: 0.0
```

```
      angular:
        x: 0.0
        y: 0.0
        z: 0.0"
```

This is just an example of how to perform a half meter movement in the x axis. If your robot has omnidirectional wheels, then you can use the y axis too.

5.6.3.2. Charge

This procedure is intended to send the robot to charge autonomously.

On 7.1.6. Robot Local Control chapter on Charge section you can find a list with all available node parameters with their description and default values.

This procedure is composed of the following steps:

First of all, the robot will change the laser safety, if the robot is equipped with this kind of laser. Then, the robot will perform a docking action using the Docker in the parameter *docker_namespace*. After that, the robot will move in the X axis as many meters as the *step_in_distance* parameter has. Then, this procedure will activate the charger relay or it will be done automatically when the contactors are pressed. Finally, the lasers can be switched to standby (to save energy) if the lasers equipped in the robot allow this option.

⚠ The robot **must** be in front of the charger before starting this procedure

---

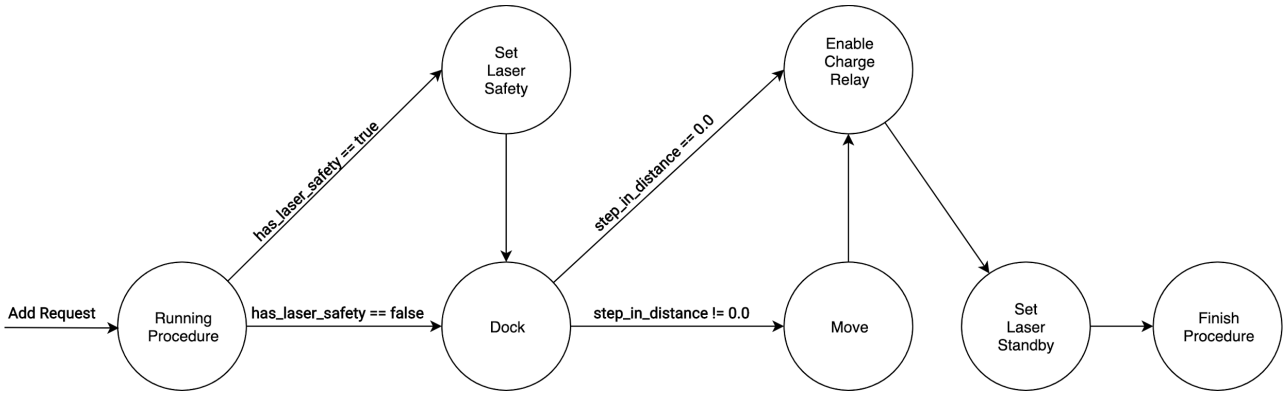


Figure 4. Charging procedure

5.6.3.2.1. Add request

In order to initiate the charge procedure you have to call the following service:

```
rosservice call
/robot/robot_local_control/NavigationComponent/ChargeComponent/add
"procedure:
  charge_station: ''"
```

If no argument is specified in *charge_station* then the procedure will use the *dock_frame* parameter as a target frame.

5.6.3.3. Uncharge

This procedure is intended to uncharge the robot autonomously.

On 7.1.7. *Robot Local Control - Uncharge* section you can find a list with all available node parameters with their description and default values.

This procedure is composed of the following steps:

First of all, the robot will activate the lasers, if they are in standby. Secondly, it will disable the charger relay so the robot will stop charging. Then, it will move backwards as many meters as the *step_back_distance* parameter has.

5.6.3.3.1. Add request

In order to initiate the uncharge procedure you have to call the following service:

```
rosservice call
/robot/robot_local_control/NavigationComponent/UnchargeComponent/add
"procedure:
  charge_station: ''"
```

5.7. Safety module

This component is intended to interact with the safety PLC of the robot via modbus.

⚠ This component is only used **when the robot is equipped with a safety PLC**. Robots without a safety PLC will not have available the topics and services listed below.

⚠ The modification of Robotnik software installation parameters could invalidate the safety and risk assessment, and could affect the guarantee of the robot.

The configurable parameters are located in the *safety_module.yaml** file in the *robot_bringup* package.

The safety module lets the robot interact with the safety PLC through modbus communication. Some information provided by the PLC is shown in the safety module topics:

- **emergency_stop** (std_msgs/Bool): Shows if the emergency stop is active.

```
rostopic echo /robot/safety_module/emergency_stop
```

- **safety_stop** (std_msgs/Bool): Shows if the emergency stop is active.

```
rostopic echo /robot/safety_module/safety_stop
```

- **state** (robotnik_msgs/State): Shows the state of the node and its frequency.

```
rostopic echo /robot/safety_module/state
```

- **status** (robotnik_msgs/SafetyModuleStatus): Shows the status of the module.

```
rostopic echo /robot/safety_module/status
```

The same way that the safety module lets read information from the PLC, it also allows writing commands to the PLC using services.

- **enable_charge** (std_srvs/SetBool): When enabling charge, write a False in the *charger_off* output and a True in the *charger_on* output. When disable charge, write a False in the *charger_on* output and a True in the *charger_off* output.

```
rosservice call /robot/safety_module/enable_charge
```

- **set_laser_mode** (robotnik_msgs/SetLaserMode): Write the new laser mode in *laser_mode_output* address.

```
rosservice call /robot/safety_module/set_laser_mode
```

- **set_safety_override** (std_srvs/SetBool): Write the request value in the *set_safety_override* output.

```
rosservice call /robot/safety_module/set_safety_override
```

- **set_to_standby** (std_srvs/SetBool): Write the request value in the *standby* output.

```
rosservice call /robot/safety_module/set_to_standby
```

5.7.1. Acoustic signals configuration

⚠ The modification of Robotnik software installation parameters could invalidate the safety and risk assessment, and could affect the guarantee of the robot. In case of acoustic signals modification, please consult the applicable safety regulation.

The acoustic signals configuration can be modified through the safety module. There are 3 possible configurations:

Scenario	Mode 1	Mode 2 (default)	Mode 3
Robot is moving	Sound off	Intermittent sound with a slower given frequency	Intermittent sound with a slower given frequency
Robot finds an obstacle	Sound off	Sound off	Intermittent sound with a given frequency

To change to **Mode 1**, just deactivate the other possible modes with the service:

```
rosservice call /robot/safety_module/set_named_output "name:
'buzzer_mode_1'           value: true"
```

```
rosservice call /robot/safety_module/set_named_output "name:
'buzzer_mode_3'           value: false"
```

To change to **Mode 3,** call the service:

```
rosservice call /robot/safety_module/set_named_output "name: buzzer_mode_3'
value: true"
```

```
rosservice call /robot/safety_module/set_named_output "name:
'buzzer_mode_1'           value: false"
```

It is also possible to force the acoustic signal to make a continuous sound for 1 second :

```
rosservice call /robot/safety_module/set_named_output "name: 'buzzer_cont'
value: true"
```

To activate/deactivate an intermittent sound use the service:

```
rosservice call /robot/safety_module/set_named_output "name: 'buzzer_inter'
value: true"
```

```
rosservice call /robot/safety_module/set_named_output "name: 'buzzer_inter'
value: false"
```

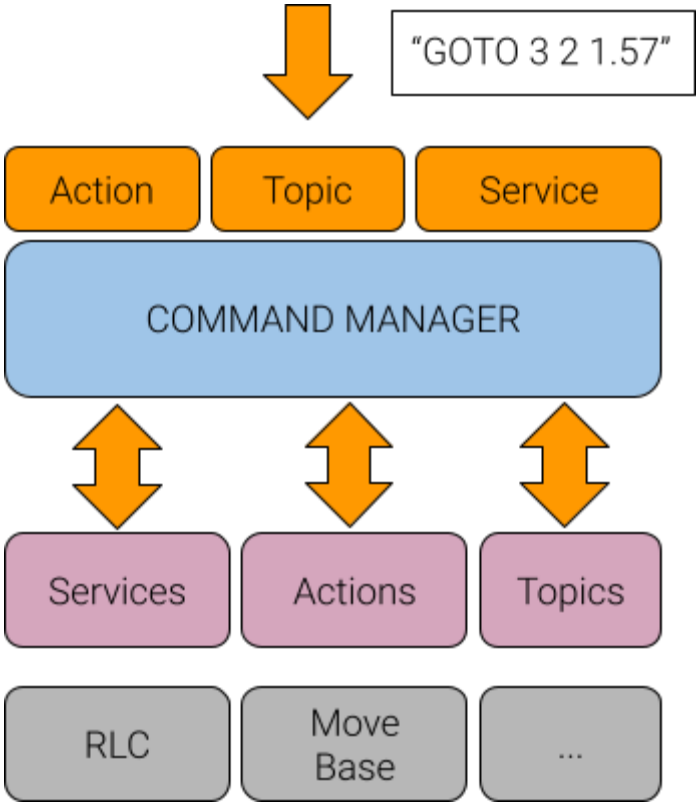
5.8. Command manager overview

The Command Manager is a set of packages that simplify user interaction with other ROS nodes. It is currently divided into four different packages:

- **robot_simple_command_manager**: This is the lowest level package. It is responsible for providing the simplified input interface in String format. For more detailed information, please refer to the Command manager chapter below.
- **robot_simple_command_sequencer**: This package allows a list of commands to be executed sequentially.
- **robot_complex_command_sequencer**: This package manages the sequences when the battery is low or the robot is in emergency.
- **robot_command_scheduler**: This packages allows to start a mission at specific time

6. Command manager

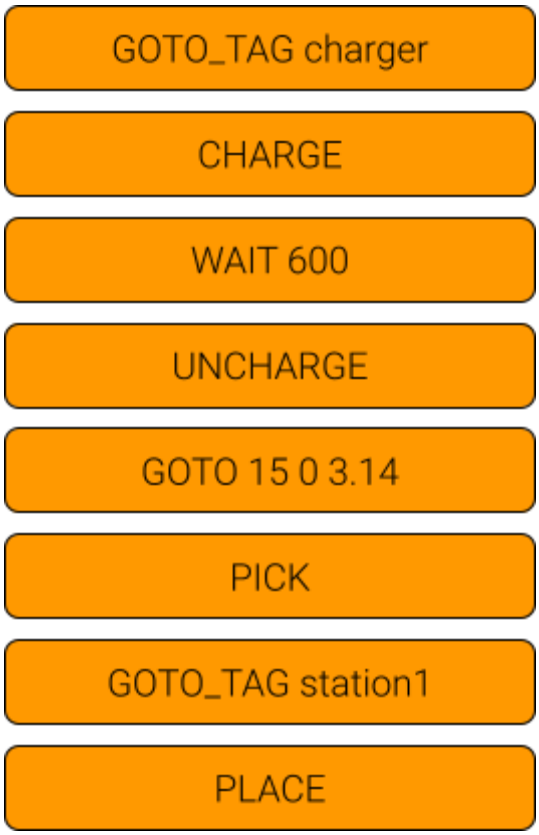
This chapter describes the use of the **robot_simple_command_manager** node and package. The robot_simple_command_manager node loads a list of handlers that allows the user to send commands using simple String encoded messages.



The main features are:

- It is intended to make the interaction **simpler** with ROS services, topics and actions.
- Interface based on **string**: TAG <arg_1> ... <arg_n>
- Each command implements a specific handler to convert data and interact with other components
- Can be commanded by using multiple ROS interfaces: services, topics and actions

The main goal of the components is to build complex routines by using a common and simple interface.
You can see an example of use in the figure below:



From now on robot_simple_command_manager node will be referred to as *Command Manager* in this document.

The folder *robot_simple_command_manager/src/robot_simple_command_manager/handlers* contains the source code for the handlers, from now on it will be referred to as *Handlers Path* in this document.

Robot Local Control procedures are sequences of actions and services developed by Robotnik, from now on it will be referred to as *Procedures* in this document.

6.1. Handlers

Python classes that parse the input String encoded messages to ROS messages and send the command to the target node. All handlers are located in the *Handlers Path* folder.

6.1.1. Types

There are different types of handlers depending on the node that connects with.

6.1.1.1. Action

These handlers connect to nodes that manage a ROS Action Server. While executing a command through an action handler, the feedback is published by the *Command Manager*.

The source code of these handlers is located in the *Handlers Path*/actions

6.1.1.2. Service

These handlers connect to nodes that manage a ROS Service Server. While executing a command through a service handler, the command is instantaneously finished.

The source code of these handlers is located in the *Handlers Path*/services

6.1.1.3. Procedure

These handlers connect to *Procedures*. While executing a command through a procedure handler, the feedback is published by the *Command Manager*.

The source code of these handlers is located in the *Handlers Path*/procedures

6.1.1.4. Subscribers

These handlers connect to topics. While executing a command through a subscriber handler, the feedback is published by the *Command Manager*. The command waits until the topic publishes the expected value.

The source code of these handlers is located in the *Handlers Path*/subscribers

6.1.2. Parameters

In order to load a Handler, two parameters are mandatory.

- **type**: Equivalent to the name of the python file where the Handler is defined concatenated to a "/" and followed by the name of the class that is defined within that file (usually the name of the file in Camel Case).

    - If the handler is defined in the robot_simple_command_manager package:

```
type: my_handler_action_interface/MyHandlerActionInterface
```

    - If the handler is defined in an external package, adding the name of the external package will be necessary:

```
type: my_package/my_handler_action_interface/MyHandlerActionInterface
```

- **namespace**: It's the namespace that the handler uses to connect to the target nodes

    - Action: the namespace equals the name of the action without the suffixes (cancel, feedback, goal and result). The handler automatically connects to that topic.

    - Service: The namespace equals to the name of the service

    - Procedure: the namespace equals the name of the procedure without the robot_local_control/NavigationComponent prefix and the suffixes (query_state, add or cancel).

6.2. Configuration

To configure the *Command Manager* there are few parameters that need to be set:

- **command_handlers**: List with the name of the desired commands.

For each command added to command_handlers, you will need to add the handler parameters (seen previously) under its namespace (The name that you used in command_handlers list).

Example:

```
command_handlers:
  - GOTO

GOTO:
  type: move_base_action_interface/MoveBaseActionInterface
  namespace: move_base
```

6.3. Usage

First it is necessary to launch the node. It will automatically load the handlers.yaml file from the config folder:

roslaunch robot_simple_command_manager robot_simple_command_manager.launch

6.3.1. Sending commands

You can send commands through topic:

```
rostopic pub /command_manager/command
robot_simple_command_manager_msgs/CommandString "command: 'GOTO 0.0 0.0
0.0'"
```

service:

```
rosservice call /command_manager/command "command: 'GOTO 0.0 0.0 0.0'"
```

or action:

```
rostopic pub /command_manager/action/goal
robot_simple_command_manager_msgs/RobotSimpleCommandActionGoal "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
goal_id:
  stamp:
    secs: 0
    nsecs: 0
  id: ''
goal:
  command:
    command: 'GOTO 0.0 0.0 0.0'"
```

For any case you will need to provide an available TAG (in this case GOTO) that has been defined in the handlers.yaml file with its required arguments.

Some aspects to take into account are:

- The **commands are not queued** in any case.
- **Sending a command does not overwrite** a command that is currently running.
- To overwrite a command, **you must first explicitly cancel it**.

You can cancel commands through topic:

```
rostopic pub /robot/command_manager/cancel std_msgs/Empty "{}"
```

service:

```
rosservice call /robot/command_manager/cancel "{}"
```

or action:

```
rostopic pub /robot/command_manager/action/cancel actionlib_msgs/GoalID
"stamp:
  secs: 0
  nsecs: 0
id: ''"
```

6.4. API

Documentation for all available handlers is shown below.

- **Description**: Short explanation of what the command does.
- **Type**: Handler type parameter of the command.
- **ROS Msg**: The handler translates the arguments into this ROS msg. The handler will be capable of connecting to any server that uses this type of messages.
- **Arguments**: Positional arguments required by the command.
- **Example**: Shows the position of the arguments and an example of the string that the user needs to send to the command manager to run the command.

6.4.1. Actions

6.4.1.1. Dock

Description	Command to move the *robot_dock_frame* to the *dock_frame* pose with an offset.	
Type	dock_action_interface/DockActionInterface	
ROS Msg	robotnik_navigation_msgs/DockAction	
Arguments	**Name**	**Type**
	dock_frame	String
	robot_dock_frame	String
	offset_x	Float
	offset_y	Float
	offset_theta	Float
Example	*TAG dock_frame robot_dock_frame offset_x offset_y offset_theta*	
	DOCK docking_station_laser robot_base_footprint 0.0 0.0 0.0	

6.4.1.2. Move

Description	Command to move the robot a distance equivalent to the passed arguments. The *y* argument is only accepted by omnidirectional robots	
Type	move_action_interface/MoveActionInterface	
ROS Msg	robotnik_navigation_msgs/MoveAction	
Arguments	**Name**	**Type**
	x	Float
	y	Float
Example	*TAG x y*	
	MOVE 1.0 0.0	

6.4.1.3. Turn

Description	Command that rotates the robot on itself according to the *theta* provided.	
Type	move_action_interface/TurnActionInterface	
ROS Msg	robotnik_navigation_msgs/MoveAction	
Arguments	**Name**	**Type**
	theta	Float
Example	TAG theta	
	TURN 1.57	

6.4.1.4. GOTO

Description	Command to move the robot with move_base based on coordinates of a map.	
Type	move_base_action_interface/MoveBaseActionInterface	
ROS Msg	move_base_msgs/MoveBaseAction	
Arguments	**Name**	**Type**
	x	Float
	y	Float
	theta	Float
	frame_id	String
Example	TAG x y theta	
	GOTO 1.0 0.5 0.0	

6.4.1.5. Wait

Description	Command to make the robot wait the *seconds* provided	
Type	wait_action_interface/WaitActionInterface	
ROS Msg	This command does not connect to a node	
Arguments	**Name**	**Type**
	seconds	Float
Example	*TAG seconds*	
	WAIT 5.0	

6.4.2. Services

6.4.2.1. Elevator

Description	Command to set the robot's elevator position.	
Type	elevator_service_interface/ElevatorServiceInterface	
ROS Msg	robotnik_msgs/SetElevator	
Arguments	**Name**	**Type**
	position	Int
Example	TAG position	
	ELEVATOR 1.0	

6.4.2.2. Save Frame

Description	Command to save a frame and its transform from fixed_frame.	
Type	save_frame_service_interface/SaveFrameServiceInterface	
ROS Msg	marker_mapping/SaveFrame	
Arguments	**Name**	**Type**
	frame_id	String
Example	TAG frame_id	
	AMCL_SAVE_FRAME robot_docking_station	

6.4.2.3. Save Map

Description	Command to save the robot map. You need to provide the name of the map file and the frame_id that it will use	
Type	save_map_service_interface/SaveMapServiceInterface	
ROS Msg	robot_local_control_msgs/SaveMap	
Arguments	**Name**	**Type**
	name	String
	frame_id	String
Example	TAG name frame_id	
	SAVE_MAP test_lab robot_map	

6.4.2.4. Set Environment

Description	Command to set the AMCL environment (map, pois, etc.)	
Type	set_environment_service_interface/SetEnvironmentServiceInterface	
ROS Msg	robot_local_control_msgs/SetEnvironment	
Arguments	**Name**	**Type**
	environment_name	String
Example	TAG environment_name	
	AMCL_SWITCH_ENVIRONMENT test_lab	

6.4.2.5. Set Pose2dStamped

Description	Command to initialize the position of the robot related to a frame_id	
Type	set_pose_2d_stamped_service_interface/SetPose2dStampedServiceInterface	
ROS Msg	robot_local_control_msgs/SetPose2DStamped	
Arguments	**Name**	**Type**
	frame_id	String
	x	Float
	y	Float
	theta	Float
Example	TAG frame_id x y theta	
	LOCALIZATION_SET_POSE robot_map 1.0 0.5 0.0	

6.4.2.6. Std SetBool

Description	Command to set bool value using ROS's standard service messages	
Type	std_setbool_service_interface/StdSetBoolServiceInterface	
ROS Msg	std_srvs/SetBool	
Arguments	**Name**	**Type**
	value	Bool
Example	TAG value	
	SET_SIGNAL true	

6.4.2.7. Std Trigger

Description	Command to trigger services using ROS's standard service messages	
Type	std_trigger_service_interface/StdTriggerServiceInterface	
ROS Msg	std_srvs/Trigger	
Arguments	**Name**	**Type**
	This command does not accept arguments	
Example	TAG	
	TRIGGER	

6.4.2.8. Switch Module

Description	Command to switch to other localization module (GPS, Amcl, Gmapping)	
Type	switch_module_service_interface/SwitchModuleServiceInterface	
ROS Msg	robot_local_control_msgs/SwitchModule	
Arguments	**Name**	**Type**
	module_name	String
Example	TAG module_name	
	LOCALIZATION_START_MODULE Amcl	

6.4.3. Procedures

6.4.3.1. Charge

Description	Command to send the robot to charge. The robot must be capable to see the charger	
Type	charge_procedure_interface/ChargeProcedureInterface	
ROS Msg	robot_local_control_msgs/ChargePetition	
Arguments	**Name**	**Type**
	This command does not accept arguments	
Example	TAG	
	RLC_CHARGE	

6.4.3.2. Goto GPS

Description	Command to send a list of coordinates to navigate using GPS	
Type	goto_gps_procedure_interface/GoToGPSProcedureInterface	
ROS Msg	robot_local_control/GoToGPSPetition	
Arguments	**Name**	**Type**
	frame	String
	max_vel	Float
	coordinates	List
Example	TAG frame max_vel coordinates	
	RLC_GOTO_GPS robot_map 1.0 []	

6.4.3.3. Goto

Description	Command to move the robot based on coordinates of a map.	
Type	goto_procedure_interface/GoToProcedureInterface	
ROS Msg	robot_local_control/GoToPetition	
Arguments	**Name**	**Type**
	x	Float
	y	Float
	theta	Float
Example	TAG x y theta	
	RLC_GOTO 0.0 2.0 1.57	

6.4.3.4. Goto tag

Description	Command to move the robot based on *tag_name* that codifies coordinates of a map.	
Type	goto_tag_procedure_interface/GoToTagProcedureInterface	
ROS Msg	robot_local_control/GoToPetition	
Arguments	**Name**	**Type**
	tag_name	String
Example	TAG tag_name	
	RLC_GOTO_TAG warehouse_1	

6.4.3.5. Move

Description	Command to move the robot a distance equivalent to the passed arguments. The *y* argument is only accepted by omnidirectional robots	
Type	move_linear_procedure_interface/MoveLinearProcedureInterface	
ROS Msg	robot_local_control/MovePetition	
Arguments	**Name**	**Type**
	x	Float
	y	Float
Example	TAG x y	
	RLC_MOVE 1.0 0.0	

6.4.3.6. Turn

Description	Command that rotates the robot on itself according to the *theta* provided.	
Type	move_angular_procedure_interface/MoveAngularProcedureInterface	
ROS Msg	robot_local_control/MovePetition	
Arguments	**Name**	**Type**
	theta	Float
Example	TAG theta	
	RLC_TURN 1.57	

6.4.3.7. Pick

Description	Command to pick a cart. It only works for those robots that has an elevator	
Type	pick_procedure_interface/PickProcedureInterface	
ROS Msg	robot_local_control/PickPetition	
Arguments	**Name**	**Type**
	pick_frame_id	String
Example	TAG pick_frame_id	
	RLC_PICK laser_cart	

6.4.3.8. Place

Description	Command to place a cart. It only works for those robots that has an elevator	
Type	place_procedure_interface/PlaceProcedureInterface	
ROS Msg	robot_local_control/PlacePetition	
Arguments	**Name**	**Type**
	This command does not require any argument	
Example	TAG	
	RLC_PLACE	

6.4.3.9. Uncharge

Description	Command to leave the charger	
Type	uncharge_procedure_interface/UnchargeProcedureInterface	
ROS Msg	robot_local_control/UnchargePetition	
Arguments	**Name**	**Type**
	This command does not require any argument	
Example	TAG	
	RLC_UNCHARGE	

6.4.4. Subscribers

6.4.4.1. Bool

Description	Command to wait for a boolean flag to be published in a topic.	
Type	bool_subscriber_interface/BoolSubscriberInterface	
ROS Msg	robot_local_control_msgs/ChargePetition	
Arguments	**Name**	**Type**
	expected_value	Bool
Example	TAG expected_value	
	EMERGENCY_STOP false	

7. Annexes

7.1. ROS Parameters

7.1.1. robotnik_move

Parameter	Description	Default value
robot_base_frame	Robot frame to perform the relative movement.	base_footprint
fixed_frame	Fixed reference to perform the relative movement.	odom
maximum_linear_velocity_x	Maximum linear velocity allowed in x.	0.2
maximum_linear_velocity_y	Maximum linear velocity allowed in y. Only used if the differential_robot is set to false.	0.2
maximum_angular_velocity_z	Maximum angular velocity.	0.2
threshold_linear_velocity_x		0.001
threshold_linear_velocity_y		0.001
threshold_angular_velocity_z		0.001
gain_linear_velocity_x	Linear x velocity gain to perform the translation.	1
gain_linear_velocity_y	Linear y velocity gain to perform the translation.	1
gain_angular_velocity_z	Angular velocity gain to perform the rotation.	1
free_goal_velocity		false
check_robot_is_moving_to_goal	Allow checking if the robot is moving during the action.	
differential_robot	Set to *false* to allow omni movements.	true

robot_moving_to_goal_averaging_time	The goal is aborted if the robot not moving during this time.	3.0
robot_moving_to_goal_delay	Period used to check if the robot is moving. Only used if *check_robot_is_moving_to_goal* is *true*.	3.0
minimum_relative_linear_velocity_between_robot_and_goal	Minimum linear velocity to consider the robot as not moving. Only used if *check_robot_is_moving_to_goal* is *true*.	0.005
minimum_relative_angular_velocity_between_robot_and_goal	Minimum angular velocity to consider the robot as not moving. Only used if *check_robot_is_moving_to_goal* is *true*.	0.02
goal_threshold_x	Maximum distance in X accepted to finish the movement successfully.	0.02
goal_threshold_y	Maximum distance in Y accepted to finish the movement successfully.	0.01
goal_threshold_theta	Maximum angle in rads accepted to finish the movement successfully.	0.01

7.1.2. Robotnik_dock

Parameter	Description	Default value
robot_base_frame	Robot frame to perform the relative docking.	base_footprint
fixed_frame	Fixed reference to perform the relative docking.	odom
maximum_linear_velocity_x	Maximum linear velocity allowed in x.	0.2
maximum_linear_velocity_y	Maximum linear velocity allowed in y.	0.2
maximum_angular_velocity_z	Maximum angular velocity.	0.2
threshold_linear_velocity_x		0.001
threshold_linear_velocity_y		0.001
threshold_angular_velocity_z		0.001
gain_linear_velocity_x	Linear x velocity gain to perform the docking.	1
gain_linear_velocity_y	Linear y velocity gain to perform the docking.	1
gain_angular_velocity_z	Angular velocity gain to perform the docking.	1
initial_maximum_distance_x	Maximum distance in x to accept a goal. It only works if *check_initial_limits* is set to true.	1
initial_minimum_distance_x	Minimum distance in x to accept a goal. It only works if *check_initial_limits* is set to true.	0.5
initial_maximum_distance_y	Maximum distance in y to accept a goal. It only works if *check_initial_limits* is set to true.	0
initial_minimum_distance_y	Minimum distance in y to accept a goal. It only works if *check_initial_limits* is set to true.	0

initial_maximum_arc	Maximum arc between the orientation of robot_base_frame and the target to accept a goal. It only works if *check_initial_limits* is set to true.	0.52
initial_maximum_yaw	Maximum difference between the orientation of robot_base_frame and the target to accept a goal. It only works if *check_initial_limits* is set to true.	0.52
check_initial_limits	Allow checking initial limits.	true
free_goal_velocity	If this option is activated (true) when the goal is surpassed, the action is finished.	false
check_robot_is_moving_to_goal	Allow checking if the robot is moving during the action.	false
publish_debug_frames	Allow publishing docking debug frames (available only in PurePursuit)	false
robot_moving_to_goal_averaging_time	The goal is aborted if the robot not moving during this time.	3.0
robot_moving_to_goal_delay	Period used to check if the robot is moving. Only used if *check_robot_is_moving_to_goal* is *true*.	3.0
minimum_relative_linear_velocity_between_robot_and_goal	Minimum linear velocity to consider the robot as not moving. Only used if *check_robot_is_moving_to_goal* is *true*.	0.005
minimum_relative_angular_velocity_between_robot_and_goal	Minimum angular velocity to consider the robot as not moving. Only used if *check_robot_is_moving_to_goal* is *true*.	0.02
goal_threshold_x	Maximum distance in X accepted to finish the docking successfully.	0.02
goal_threshold_y	Maximum distance in Y accepted to finish the docking successfully.	0.01
goal_threshold_theta	Maximum angle in rads accepted to finish the docking successfully.	0.01
tf_timeout	Maximum time to set the action as	0.01

aborted if the tf is too old.

7.1.3. Move base

Parameter	Description	Default value
controller_frequency	The rate in Hz at which to run the control loop and send velocity commands to the base.	20
controller_patience	How long the controller will wait in seconds without receiving a valid control before space-clearing operations are performed	15
planner_frequency	The rate in Hz at which to run the global planning loop. If the frequency is set to 0.0, the global planner will only run when a new goal is received or the local planner reports that its path is blocked	0
planner_patience	How long the planner will wait in seconds in an attempt to find a valid plan before space-clearing operations are performed.	5
conservative_reset_dist	The distance away from the robot in meters beyond which obstacles will be cleared from the costmap when attempting to clear space in the map. Note, this parameter is only used when the default recovery behaviors are used for move_base.	0
recovery_behavior_enabled	Whether or not to enable the move_base recovery behaviors to attempt to clear out space.	true
clearing_rotation_allowed	Determines whether or not the robot will attempt an in-place rotation when attempting to clear out space. Note: This parameter is only used when the default recovery behaviors are in use, meaning the user has not set the recovery_behaviors parameter to anything custom.	true

shutdown_costmaps	Determines whether or not to shutdown the costmaps of the node when move_base is in an inactive state	false
oscillation_timeout	How long in seconds to allow for oscillation before executing recovery behaviors. A value of 0.0 corresponds to an infinite timeout.	0
oscillation_distance	How far in meters the robot must move to be considered not to be oscillating. Moving this far resets the timer counting up to the ~oscillation_timeout	0.5
max_planning_retries	How many times to allow for planning retries before executing recovery behaviors. A value of -1.0 corresponds to an infinite retries.	1

7.1.4. Robot Local Control - GoTo

Parameter	Description	Default value
desired_freq	Desired frequency of the node.	5.0
action_namespace	Namespace of the move_base server	move_base
global_frame	Global frame used to perform move_base goals	robot_map
base_frame	Base frame used to perform move_base goals	robot_base_footprint
has_safety_laser	Flag to indicate if the robot has safety laser. If it has, it will change the safety area before performing the move_base goal	false
default_yaw_tolerance		
yaw_toleramce	Move base goal yaw tolerance	0.1
xy_tolerance	Move base goal x and y tolerance	0.15
clear_costmaps_before_send_goal	Flag to clear costmap every time that a procedure is added	false
local_planner_namespace	Namespace of the local planner used by move_base	TEBLocalPlanner

7.1.5. Robot Local Control - Move

Parameter	Description	Default value
desired_freq	Desired frequency of the node.	5.0
action_namespace	Namespace of the robotnik_move server	move

7.1.6. Robot Local Control - Charge

Parameter	Description	Default value
desired_freq	Desired frequency of the node.	5.0
docker_namespace	Namespace of the docker server	pp_dock
move_namespace	Namespace of the robotnik_move server	move
has_safe_charge	Flag to indicate if the robot has safety charge. If it has, it will manage the necessary I/O of the PLC to enable the charge	false
has_laser_safety	Flag to indicate if the robot has safety laser. If it has, it will change the safety area before performing the docking and move goals	false
dock_frame	Target frame that the robot uses to perform the dock action	robot_docking_station_laser_filtered
robot_dock_frame	Frame of the robot that will finish in the same position and orientation that the dock_frame, taking into account the provided offset	robot_base_footprint

dock_offset_x	Distance applied to the dock_frame position to perform the docking	-0.6
step_in_distance	The distance that the robot is going to move in x-axis after the docking. If it equals to 0.0, then no move action is done	0.0
set_laser_to_standby	If it is true then the lasers will be in standby after docking	false

7.1.7. Robot Local Control - Uncharge

Parameter	Description	Default value
desired_freq	Desired frequency of the node.	5.0
move_namespace	Namespace of the move server	move
has_laser_safety	Determines whether the robot has safety lasers equipped or not	false
laser_mode_at_finish	Desired laser mode at the end of the procedure	-
rotation	Angle, in radians, that the robot will rotate	0.0
step_back_distance	Distance, in meters, that the robot will move backwards	0.0

7.1.8. Safety module

Parameter	Description	Default value
desired_freq	Desired frequency of the node.	5.0
address_registers/laser_mode_output	Safety PLC register address where the laser mode is set.	2001
address_registers/current_speed	Safety PLC register address where the current speed is set. Only used if set_speed_feedback_to_safety_module is true.	2002
inputs/emergency_stop	Safety PLC input that shows if emergency stop is active.	228
inputs/ready_to_charge	Safety PLC input that shows if the robot is located in docking station.	233
inputs/charging_contactor	Safety PLC input that shows if charging contactor is active.	234
inputs/auto_mode	Safety PLC input that shows if the auto mode is active.	230
inputs/emergency_mode	Safety PLC input that shows if the emergency mode is active.	231
inputs/manual_mode	Safety PLC input that shows if the manual mode is active.	232
inputs/laser_mode_std	Safety PLC input that shows if the laser are in standard mode.	237
laser_mode_docking_station	Safety PLC input that shows if the laser are in docking station mode.	240
inputs/laser_mode_docking_cart	Safety PLC input that shows if the laser are in docking cart mode.	238
inputs/laser_mode_cart	Safety PLC input that shows if the laser are in cart mode.	239

inputs/safety_overrided	Safety PLC input that shows if the safety is overridden. Only in manual mode.	235
inputs/safety_stop	Safety PLC input that shows if the safety stop is active.	225
inputs/standby	Safety PLC input that shows if the laser is in standby.	238
outputs/safety_override	Safety PLC output to override safety.	3
outputs/charge_on	Safety PLC output to start charging.	9
outputs/charge_off	Safety PLC output to stop charging.	10
outputs/standby	Safety PLC output to set lasers to standby.	13
outputs/watchdog_signals	Safety PLC output to send watchdog signals.	Empty list of integers
watchdog_signals_frequency	Frequency of the watchdog signals.	2.0
set_speed_feedback_to_safety_module	Allow sending robot speed to the safety PLC.	false

7.2. ROS Messages

7.2.1. Move.action

```
# goal
geometry_msgs/Pose2D goal
geometry_msgs/Twist maximum_velocity
---
# result
bool success
string description
---
# feedback
geometry_msgs/Pose2D remaining
geometry_msgs/Twist current_velocity
```

7.2.2. Dock.action

```
# goal
string dock_frame
string robot_dock_frame
geometry_msgs/Pose2D dock_offset
geometry_msgs/Twist maximum_velocity
---
# result
bool success
string description
---
# feedback
geometry_msgs/Pose2D remaining
geometry_msgs/Twist current_velocity
```

RB-KAIROS+

Maintenance manual

Revision v.1.2

Contents

1. Introduction

Welcome to the RB-KAIROS+ Maintenance manual. This document describes the maintenance instructions for the robot, including electrical and mechanical components, default configuration and recycling of the robot.

This document is dedicated to the maintenance operator, with a medium knowledge level of electronics and mechanics.

Please, pay close attention to safety warnings before operating the RB-KAIROS+ or performing any maintenance action.

For further information, you can consult the following documents:

- RB-KAIROS+ User manual
- Developer manual
- Control interface manual
- Appendices

1.1. Contact

Robotnik headquarters:
SPAIN
Robotnik Automation S.L.L
Ronda Auguste y Louis Lumière, 8
46980 P. Tecnológico, Paterna
Valencia (España)

If you have any doubt or problem with your robot, please contact us by sending an email to support@robotnik.es, indicating the serial number of your robot.
We are constantly improving and upgrading our products and services. Any feedback from users is welcome.

NOTE: You can get in contact with us either in English or Spanish.

1.2. Robot Types

RB-KAIROS+ is a mobile manipulator that has always integrated a collaborative robotic arm from Universal Robots. With six points of articulation and a wide range of flexibility, these robotic arms are designed to mimic the range of motion of a human arm.

Depending on the reach and the payload of the arm, there are four different options that can be installed: UR3e, UR5e, UR16e and UR10e.

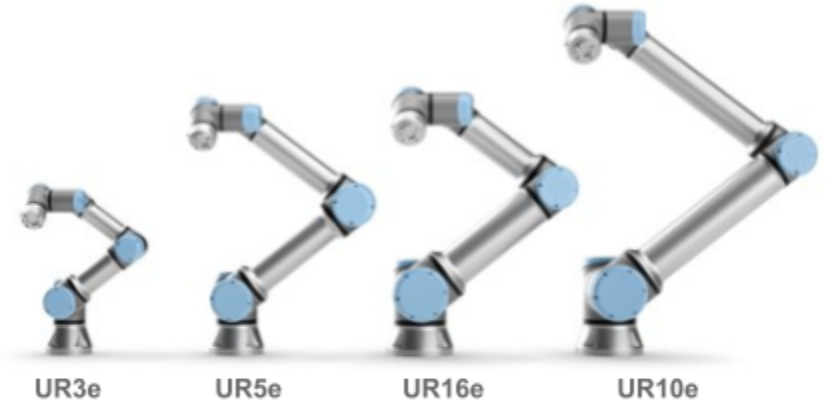


Figure 1 - UR arm family

According to the chosen arm, the RB-Kairos series is composed by: RB-KAIROS+ 3e, RB-KAIROS+ 5e, RB-KAIROS+ 16e and RB-KAIROS+ 10e.

This manual is applicable to all the RB-KAIROS+ series.

2. Safety Warnings

While reading this guide you may find some warning blocks. These include important information related with common errors or safety concerns that the final user must know.

In this chapter you will find the most significant and specific safety messages for robot maintenance. Please refer to the RB-KAIROS+ user manual for reading a complete and general list of safety messages.

2.1. Safety concept

WARNING

"Warning" description includes information of an error and some possible solutions.

FORBIDDEN

"Forbidden" description includes information about the forbidden situation and how to avoid it.

MANDATORY

"Mandatory" description includes information to solve forbidden situations.

2.2. Safety instructions

Do not modify installation parameters of the robot. You could damage the robot or injure someone.

Modifications or add-ons of hardware, and modifications of Robotnik software installation parameters could invalidate the safety and risk assessment.

The access to the robot electronics must always be done by disassembling the two side covers first and, if necessary, the second central cover. The central cover is the last element to remove. Be careful removing it. Incorrect use could damage the robot.

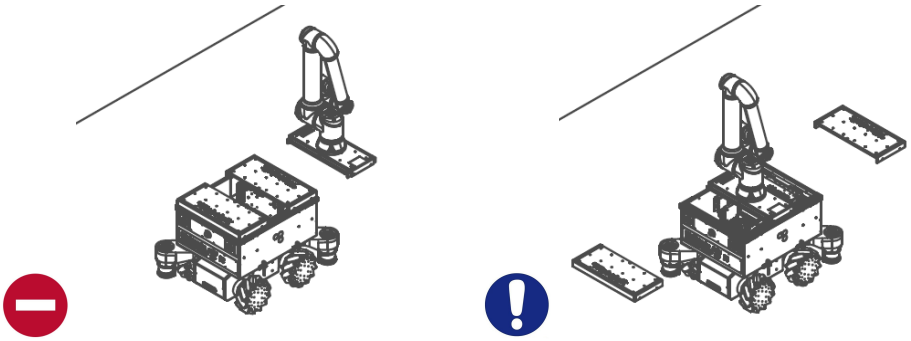


Figure 2 - Top cover disassembling order

Keep the emergency stop pussed if there is an operator close to the robot.

If there is evidence of battery malfunction, please follow the steps listed below. Use personal protective equipment, such as gloves, safety glasses and laboratory coat.

- If batteries show evidence of thermal runaway failure, be very cautious because the gasses may be flammable and toxic and failure modes can be hazardous.
- Disconnect the battery. Do not disassemble or break it.
- Remove the battery from the robot.
- Place the battery in a metal or other container away from combustibles.

In case of battery leakage entering into one's eye, rinse with water and look for medical advice immediately. Do not rub the eye.

3. Working mode: Maintenance

RB-KAIROS+ has three different working modes of the mobile base.
The base working modes are independent of the working modes of the robotic arm.
The base working mode affects safety elements and conditions.

For a Maintenance process, it is recommended to use the Base Maintenance Mode of the robot. When the Manual Mode is selected, the safety lasers are always disabled, independently of the laser key selector.

For safety reasons, the base and the arm will only move under specific conditions.

3.1. Conditions for base movement

The safe movement of the robot will be enabled if the following conditions are met:

- All the emergency buttons are released and the RESTART button has been pressed.
- All the internal components are ready and powered.
- The arm is located or moving inside the robot's footprint.

3.2. Conditions for arm movement

The safe movement of the arm will be enabled if the following conditions are met:

- All the emergency buttons are released and the RESTART button has been pressed.
- The arm is powered on and initialized.
- The arm brakes are released.

The "safe stop" in maintenance mode is activated when the emergency button is pressed. The "emergency stop" will be triggered in the arm, and the running program will stop, without possibility of being resumed.

For a detailed explanation of the stop types please refer to the User Manual of the arm.

WARNING

The robot and the arm will move freely in any case if the key is in the “MAINTENANCE” mode as described in Rear panel section .

While moving RB-KAIROS+ in maintenance mode, you must pay attention to the robot and the environment. Otherwise, you could damage the robot or injure someone.

This is only recommended for experienced users.

3.3. Controls

In manual mode, the base can be driven with the dualshock gamepad.

Figure 3- Gamepad Operation

For further information, please refer to the RB-KAIROS+ user manual.

4. Maintenance

WARNING

Switch off the power of the robot before starting any maintenance task, specially in order to manipulate the wheels or any other mechanical part, or access the electrical cabinets. Otherwise you can damage yourself or the robot.

4.1. Maintenance schedule

The following table summarizes all the elements that need maintenance and the periodicity of this maintenance.

	Often	Every 6 month	Observations
Screws	Check they are not loosen.	Tighten nuts if it is necessary	
Wheels		Visual control of the wheel.	Replace them when the mecanum wheel rollers or rubber tire need it. Dented, cracked, dry or sticky rubbers are signs that indicate the need of maintenance.
Wheel wires		Visual control	Replace them when necessary. For further information, check Motor wheel chapter.
Outer wires		Visual control of wear.	If wear appears, temporarily protect them with a shrink tube and replace them as soon as possible.
Emergency stop button	Check regularly that the emergency stops work properly.		Change the button if it does not work well.
Battery	Control Batteries Voltage, do not let the batteries get fully discharged.	Check battery autonomy.	Recharge as soon as possible if fully discharged.

Fans	Check regularly fan filters.	Check the fans work correctly.	Clean the four filters when needed. Change the fan if it does not work well. To clean the filters it is necessary to remove the external covers by hand, remove the sponge and wipe it dry. To replace the fans it is necessary to remove the 4 screws that hold it and disconnect it from the terminals. To access the disassembly it is necessary to open the upper covers of the robot.
Optical sensors	Clean the optics of the sensors regularly to avoid false obstacles.		Clean the optics with a dry and soft cloth. For further information consult the manuals of the devices.
Safety Lasers	Clean the surface of the lasers to avoid false detection.		Clean the optics with a dry and soft cloth. For further information consult the manuals of the devices.
Led lights		Check the correct deployment of the lights, specially for safety functions as moving or safety stop.	
Dualshock gamepad	Keep the battery charged	.	
External reflectors and markers	Check regularly the cleanliness of the reflectors		Clean the external reflectors and markers when needed.
Floor	Check regularly the cleanliness and integrity of the floor at the action area.		Avoid dirty or wet floor that can cause wheel sliding. Avoid indetectable obstacles or unsuitable irregularities on the robot path.

For further information about robot maintenance schedule, please contact with Robotnik support department. In the following chapters you will find more information about the maintenance of specific robot components.

4.2. Restore Factory Defaults

There are several components on the electronic board that need a specific configuration to obtain a proper robot behavior: safety PLC, Motor drives, CPU…

The configuration of these components can be modified by an experienced user or maintenance personnel at his own risk. Due to the importance of these components, it is mandatory to make a backup of the default configurations before modifying them.

For this purpose, the specific process for each component is as follows:

- **Motor drives**: Depending on the brand of drives your robot has, a different software is used. If the drive is Ingenia, the software is MotionLab3 1.5.1, if the drive is AMC the software is Driveware 7.4.2. In order to connect to the first one, you must connect to the Peak-CAN USB connected to the CPU and click “Save all” to store the current configuration in a local folder, and “Store all” to save the changes made. When the software used is Driveware 7.4.2., it is only necessary to connect to the corresponding driver, click on "Upload" to load the written data to the PC, and "Save as" in a local folder.
- **Safety PLC:** The software used is Safety Designer 1.13.1., it will be necessary to connect to the program, click on "Upload" to load the written data to the PC and "Save as" in a local folder.
- **CPU**: For further information, please refer to the Developer manual.

In case you modify any parameter of any device incorrectly or you want to return to the Robotnik standard configuration, it will be necessary to reload these default settings in the corresponding device.

For further information, please refer to the official manual of the specific component or software.

WARNING

The modification of Robotnik software installation parameters could invalidate the safety and risk assessment, and could affect the guarantee of the robot.

4.3. Component maintenance

4.3.1. Battery

The battery is composed of 3.2V LiFePO4 cells and a protection circuit module. With this technology the robot should be able to operate between 3 and 10 hours. The standard battery is 30Ah@48VDC, if you have a different type of battery please contact the Robotnik support department.

The robot circuit is powered when the general switch is ON. The control DC/DC converter, which supplies power to the different devices of control, is powered at the same time.

The battery must be kept clean and dry in order to avoid escape currents. Check the wear out of the battery wires to prevent short circuits.

Recharge the batteries as soon as possible if they get fully discharged. Keeping the voltage low for a long time will greatly reduce the life cycles.

There are three important values for the levels of the battery that can be checked with a tester or via software. These values are:

- Charging voltage: Is the level in the robot during the charging process. Normal values are around 57.6V (16 cells per 3.6V each cell) Note that after charging, the voltage will drop fast to "Full charge voltage" even without using the battery).
- Full charge voltage: Is the level in the robot with the battery full charge. Normal values are around 53.6V (16 cells per 3.35 V each cell).
- Full discharge voltage: Is the level in the robot with the battery empty. Normal values are around 46.4V (16 cells per 2.9V each cell).

IMPORTANT: The cells can drop up to 2.5V, but it is not recommended. When the battery level is very low, the BMS can shut down easily if any power peak is required.

If there is evidence of battery malfunction, please follow the steps listed below. Use personal protective equipment, such as gloves, safety glasses and laboratory coat.

- If batteries show evidence of thermal runaway failure, be very cautious because the gases may be flammable and toxic and failure modes can be hazardous.
- Disconnect the battery. Do not disassemble or break it.
- Remove the battery from the robot.
- Place the battery in a metal or other container away from combustibles.

In case of battery leakage entering into one's eye, rinse with water and look for medical advice immediately. Do not rub the eye.

WARNING

The battery may get hot, ignite or explode in case of electrical or mechanical abu order to avoid these situations, follow the next instructions:

- Do not short-circuit the battery.
- Do not mechanically harm or disassemble the battery.
- Do not expose the battery to extreme temperatures, according to its manu
- Do not allow the battery to get wet.
- Do not charge the battery outside the robot

4.3.1.1. Access to the battery

In order to take out the battery from the robot, first open the rear gate by removing the 4 screws that keep it closed. Then, unplug the power supply connector (red and blue wires) and the charge connector (yellow wire).

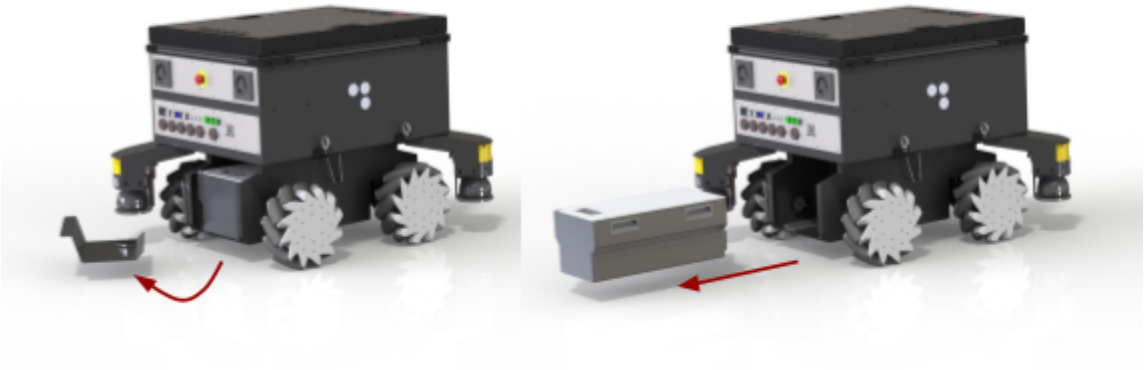


Figure 4 - Battery gate

4.3.2. Motor wheel

The most important thing to check on the motor wheels is the wire located inside the robot. This wire must be kept in good condition, and protected if the external cover is damaged. There are three 48V power wires, two 5VDC power and three Hall Effect signals. If they are short-circuit, the motor and the driver can be damaged. To access it, open the rear gate as explained in the previous chapter " Battery".

4.3.2.1. Wheel disassembly instructions

Wheels are attached to the hub via 6 hexagonal head screws, as shown in the next figure.

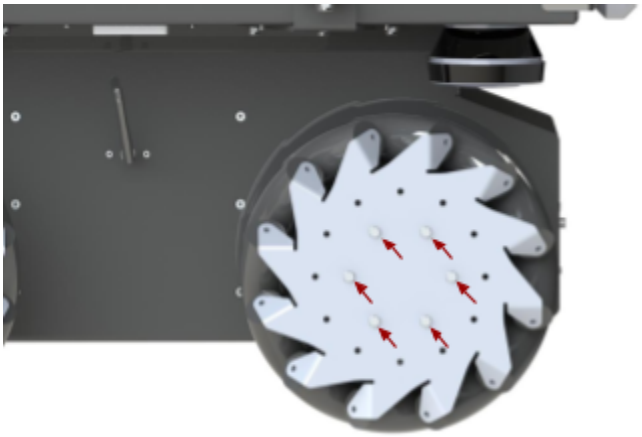


Figure 5 - Wheel screws for disassembly

The **tightening torque** recommended to the screws for mecanum wheels is **4.5 N.m**.

4.3.2.2. Motor drives

The drivers are programmed at Robotnik with specific settings for each motor. The serial identifier is the default one (63), but each driver has its own CAN bus address (1, 2, 3 and 4). DO NOT change them from one motor to another.
You may notice that when the velocity is zero, the rear wheels are braked but the front ones are free. This behaviour is applied in order to reduce the power consumption during this phase.

The computer sends CAN messages to move the robot, and they are different from the left side (1 & 2) and right side (3 & 4). Driver 1 is the only one with the Can bus resistor installed.

WARNING

Any change on the Motor drives could invalidate the safety assessment.

4.3.3. Electronic components

To access the electronic board you have to remove two of the three top covers of the base. Please always remember not removing the central cover. You should only remove that cover if you need to disassemble the robotic arm.

In order to remove the covers, remove the needed screws: there are 6 screws on the top and 6 more on each side, as shown in the image below:

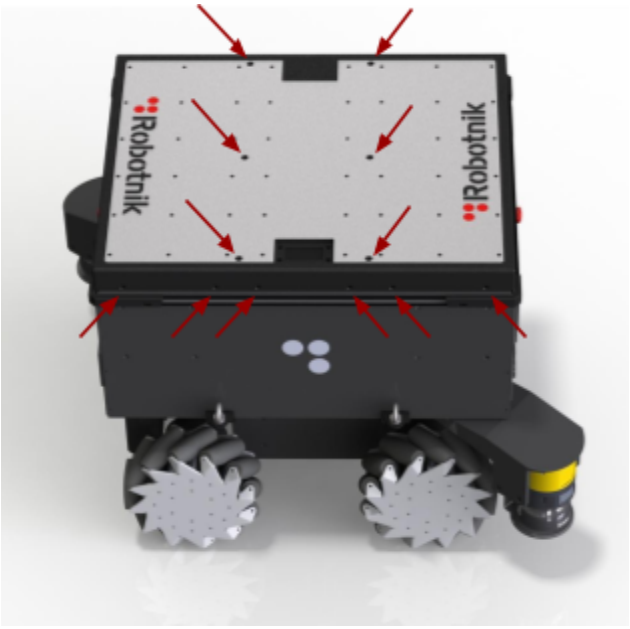
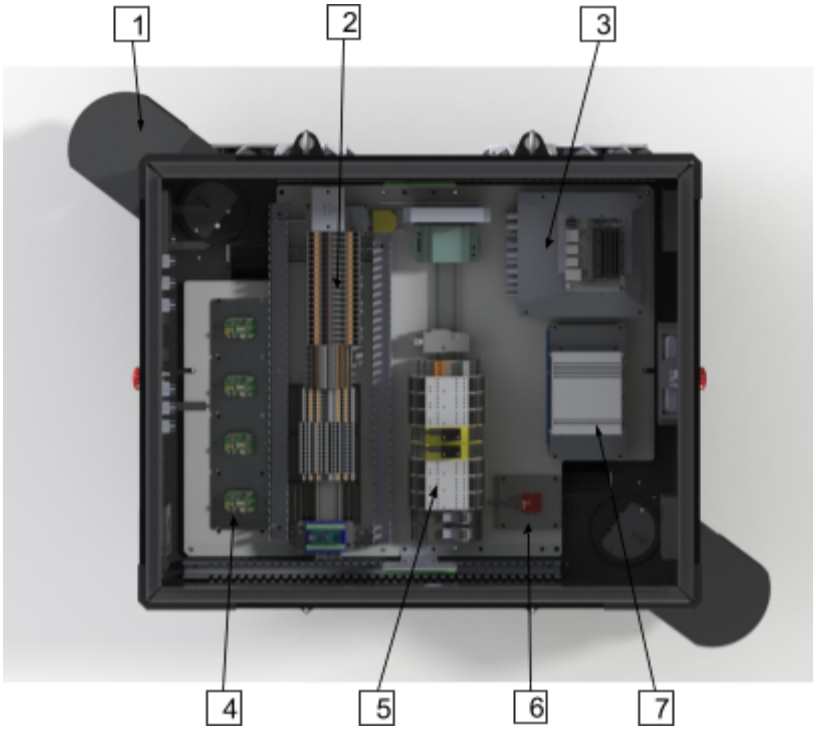


Figure 6 - Screws for electronic board access

The next figure shows the components of the default RB-KAIROS+ electronic board:



1. 2D Laser
2. Fuses and terminals
3. Platform CPU
4. Motor drives
5. Safety modules
6. IMU
7. Router and switch

Figure 7 - Components of RB-KAIROS+ electronic board

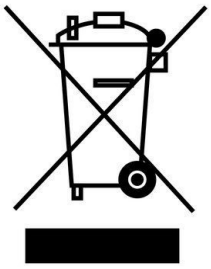
5. Recycling of the robot

Robotnik Automation SL is committed to providing quality products in an environmentally sound manner. For this reason, Robotnik Automation SL continuously improves the design processes of its products to minimize the negative impact in their life cycle.

Design for recycling has been incorporated into this product:

- The number of used materials has been kept to the minimum, while ensuring proper functionality and reliability.
- Dissimilar materials have been designed to separate easily.
- Fasteners and other connections are easy to locate, access, and remove using common tools.
- High-priority parts have been designed so that they can be accessed quickly, providing efficient disassembly and repair.

The packaging materials for this device have been selected to provide maximum protection at the lowest possible cost, while attempting to minimize environmental impact and ease the recycling process.

	**European Union**: under the directive 2002/96 / EC of the European Union regarding waste and / or electronic equipment, with a rigorous date since August 13, 2005, products classified as "electrical and electronic equipment" cannot be deposited in usual containers of your municipality, equipment manufacturers electronic, are required to take care of these products at end of his life period. Use the public collection system to return, recycle or treat them in accordance with local regulations.

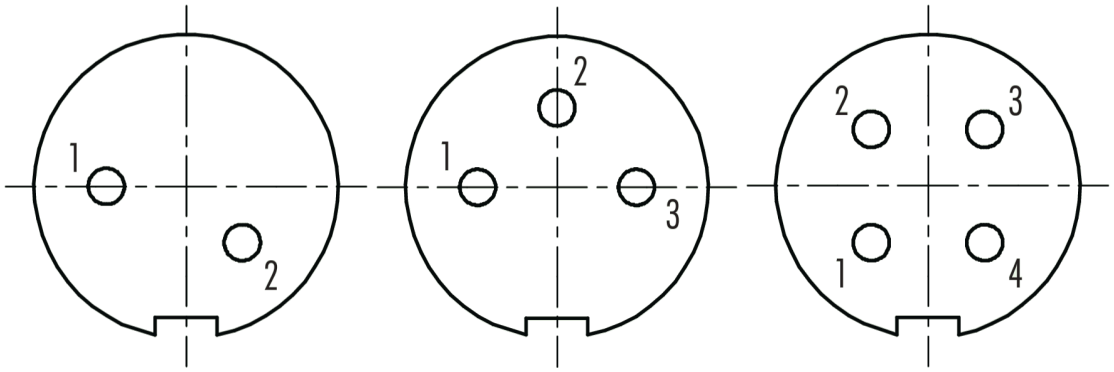
6. Guarantee

The warranty of the products offered will be one year from the date of delivery of products, in case of subsystems provided by thirty parts i.e. the Arms, the period provided by the manufacturer. It includes replacement parts and delivery of the parts, it does not include the mounting of the damaged parts at client facilities. Robotnik will provide technical support for hardware and software via e-mail, phone and previously scheduled Skype telemeeting (in English or Spanish) during Robotnik working hours, for the warranty period. We provide free access to the software updates to our clients even after the warranty period.

7. Annexes

7.1. Pinout robot interfaces

The standard RB-KAIROS+ has the following pinout connections:



	CURRENT	CONNECTOR	PINOUT	DESCRIPTION
BAT	2 x 3A	4 pin	1, 3	GND BAT
			2, 4	+BAT
24VDC	2A	3 pin	1	GND 24V
			2	+24VDC
			3	NC
12VDC	2A	2 pin	1	GND 12V
			2	+12VDC

The compatible commercial connectors are these:

- Straight Connectors:
  - 2 pin: 99-0401-00-02

    - 3 pin: 99-0405-00-03
    - 4 pin: 99-0409-00-04
- Angled connectors:
    - 2 pin: 99-0401-70-02
    -
    - 3 pin: 99-0405-70-03
    - 4 pin: 99-0409-70-04

If you have ordered a different configuration, please refer to the Appendices chapter.

Robotnik

RB-KAIROS+

User manual

Revision v.1.3

Contents

1. Introduction

Welcome to the RB-KAIROS+ user manual. This document describes the main aspects of the robot, including components, basic operation and safety specifications.

This document is dedicated to the daily operator of the robot, with basic knowledge of informatics and programming (taking in account Polyscope software).

Please, pay close attention to safety warnings before unpacking and operating the RB-KAIROS+.

For further information, you can consult the following documents:

- RB-KAIROS+ maintenance manual
- Developer manual
- Control interface manual
- Appendices

1.1. Contact

Robotnik headquarters:
SPAIN
Robotnik Automation S.L.L
Ronda Auguste y Louis Lumière, 8
46980 P. Tecnológico, Paterna
Valencia (España)

If you have any doubt or problem with your robot, please contact us by sending an email to support@robotnik.es, indicating the serial number of your robot.
We are constantly improving and upgrading our products and services. Any feedback from users is welcome.

NOTE: You can get in contact with us either in English or Spanish.

1.2. Robot Types

RB-KAIROS+ is a mobile manipulator that can integrate a collaborative robotic arm from Universal Robots. With six points of articulation and a wide range of flexibility, these robotic arms are designed to mimic the range of motion of a human arm.

Depending on the reach and the payload of the arm, there are four different options that can be installed: UR3e, UR5e, UR10e, UR16e.

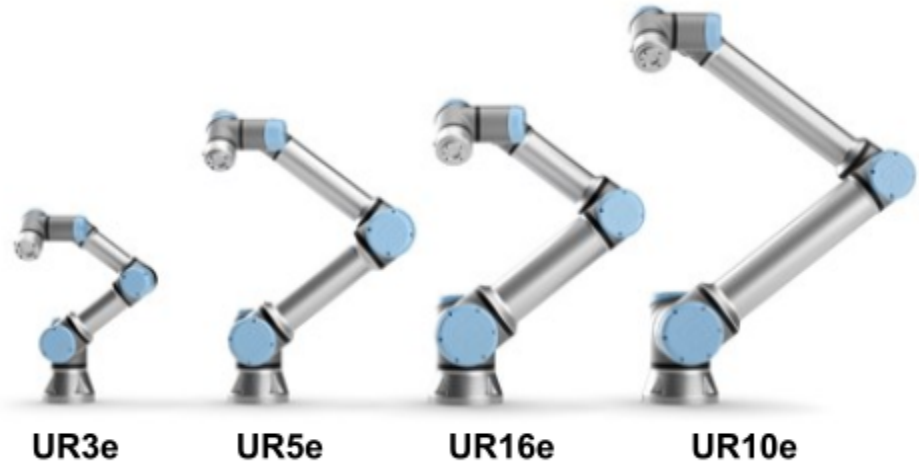


Figure 1 - UR arms

According to the chosen arm, the RB-KAIROS+ series is composed by: RB-KAIROS+ (no arm), RB-KAIROS+3e, RB-KAIROS+5e, RB-KAIROS+16e and RB Kairos+10e.

This manual is applicable to all the RB-KAIROS+ series.

1.3. Intended use

RB-KAIROS+ is a collaborative mobile manipulator, able to work in indoor industrial environments. Its software and hardware are fully prepared to mount the Universal Robots arm and thus turn the robotic arm into a mobile manipulator that can operate in large areas.

RB-KAIROS+ is extremely useful for industrial applications such as:

- Pick, transport and place
- Logistics, warehouses and other intra-logistics applications
- Industrial mobile manipulation, such as part feeding, metrology, quality control, packaging, cleaning, polishing, screwing, drilling, etc.

RB-KAIROS+ is designed with a robust steel design and can carry up to 250Kg. Thanks to its safety laser scanners and signals, the robot can safely share the workspace with the operators.

The mobile platform has omnidirectional kinematics based on 4 high power wheel motors, allowing optimization of its trajectories.

RB-KAIROS+ can navigate autonomously and can be configured with a wide range of sensors and components found within the UR+ ecosystem. The mobile base is able to go to a predefined point, move relatively to its current position, go to charge or dock to a predefined marker. The system is easy to manage either by UR programming interface or ROS framework.

Without doubt, it is an excellent way to improve the productivity of any factory.

2. Main components

The next figure shows the main external parts of the default robot:

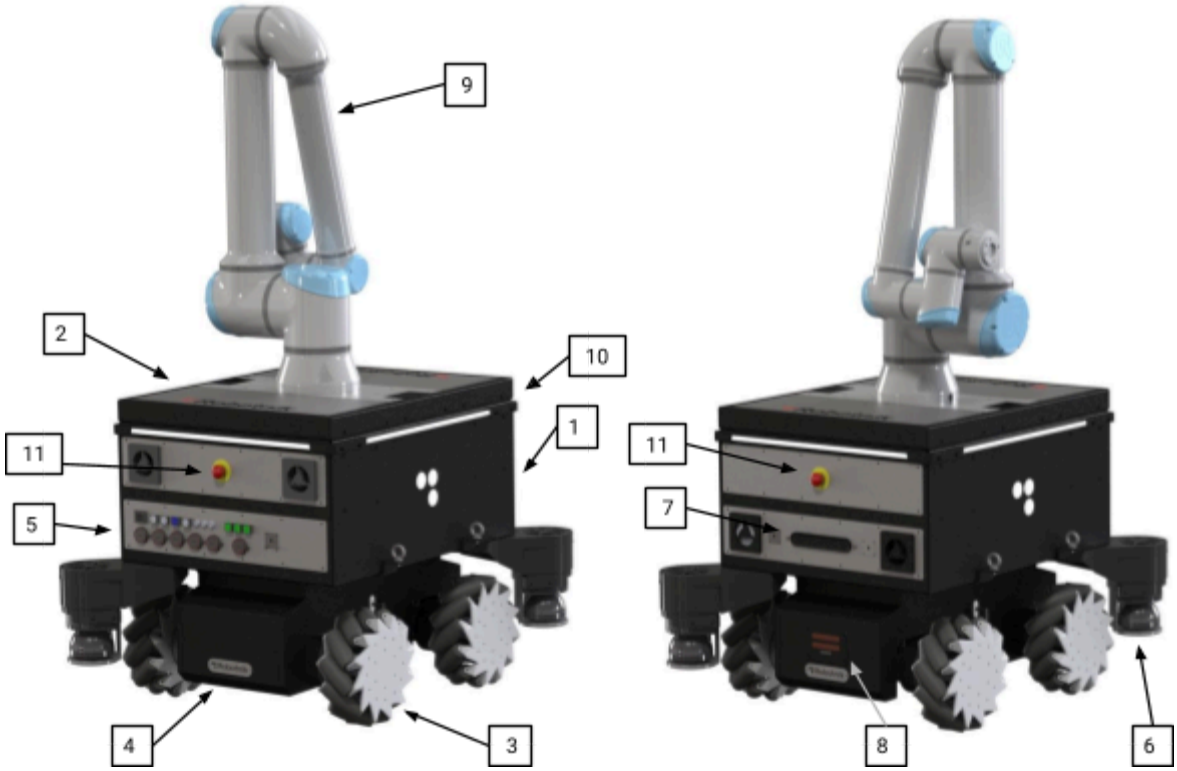


Figure 2 - RB-KAIROS+ main components

1. Base main chassis
2. Top Covers
3. Motor wheel
4. Back Gate
5. Rear panel
6. Laser sensor
7. 3D Camera
8. Charger contacts
9. Arm
10. Light indicators
11. Emergency stops

Rear panel:

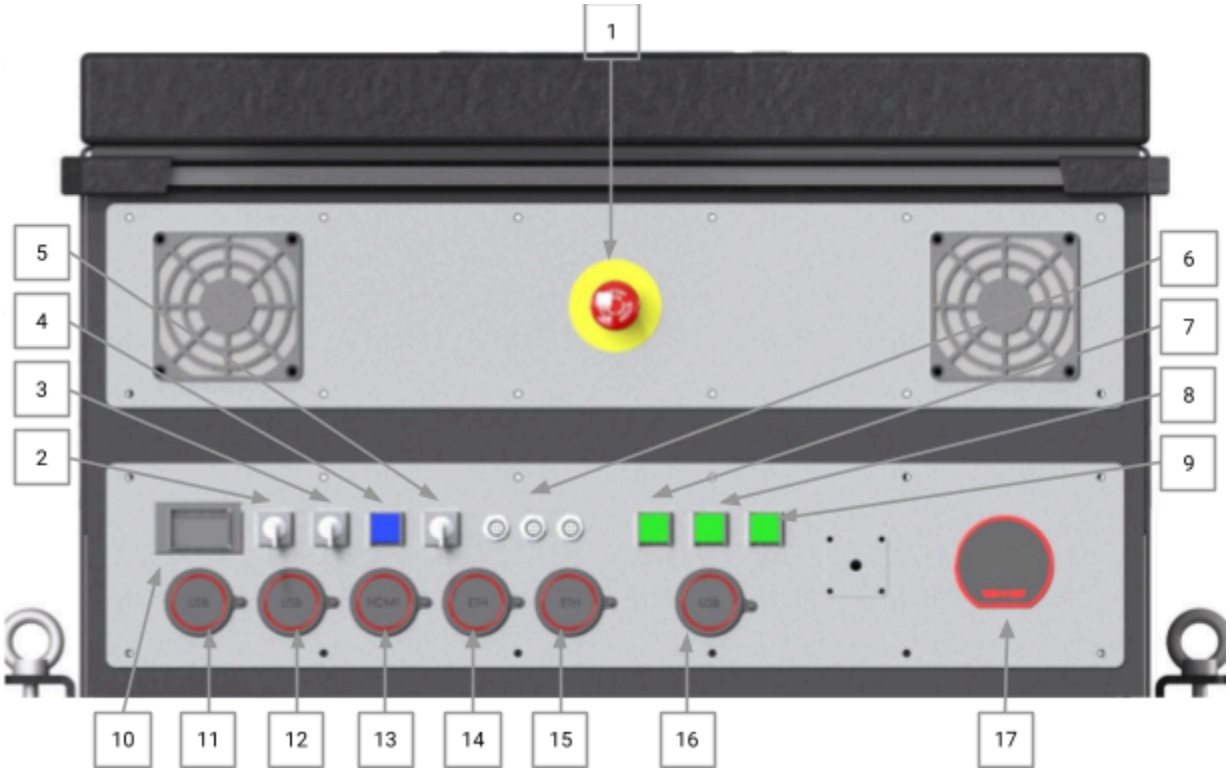


Figure 3 - RB-KAIROS+ rear panel

1. Emergency stop button
2. Main power selector (On/Off)
3. Working mode key selector
4. Restart button
5. Laser enabled/muted key selector
6. Power connectors : 12V, 24V, VDC BAT
7. PC Robot indicator
8. CPUs power off button
9. PC UR arm indicator
10. Manual charger connector
11. USB 2.0 port to the base CPU
12. USB 2.0 port to the base CPU
13. HDMI port to the robot CPU
14. Wan Ethernet port
15. LAN Ethernet port
16. HDMI + USB port to the UR arm
17. Battery status display

Emergency stop buttons:

In case of any emergency, the RB-KAIROS+ has 2 available emergency stop buttons: one on the front and one on the back. Push any of them to stop the vehicle. Release it and rearm the robot to get back to normal operation.

Optionally, as shown in the image on the right, the robot can have a remote emergency stop device installed.

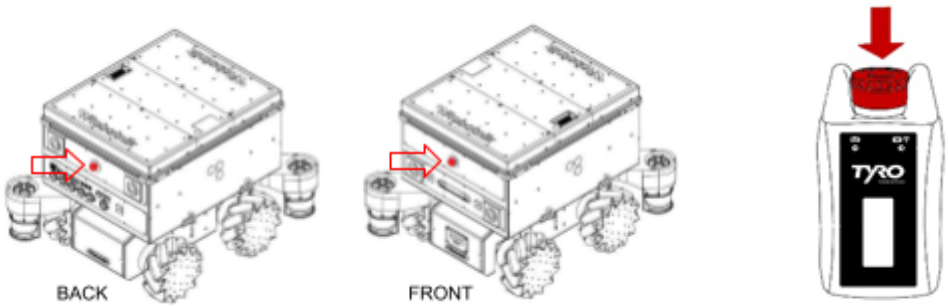


Figure 4 - RB-KAIROS+ emergency stops

Dualshock gamepad:

Figure 5 - Dualshock controller

Manual charger:

Figure 6 - Manual charger

For further information, please consult the “Robot components” chapter.

3. Safety Warnings

3.1. Safety concept

While reading this guide you may find some warning blocks. These include important information related to common errors or safety concerns that the final user must know.

WARNING

"Warning" description includes information of an error and some possible solutions.

FORBIDDEN

"Forbidden" description includes information about the forbidden situation and how to avoid it.

MANDATORY

"Mandatory" description includes information to solve forbidden situations.

3.2. General safety instructions

The robot must not be used when the two side covers are removed. You could damage the robot or injure yourself.

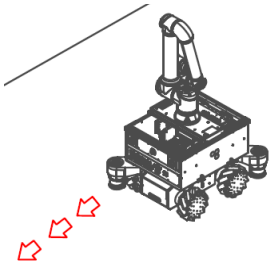


Figure 7 - Forbidden robot moving with covers removed

The robot platform should not be moved when the robot is connected to the manual charger. You could damage the robot.

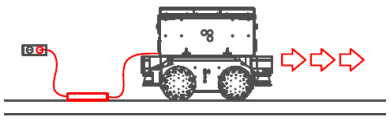


Figure 8 - Forbidden robot moving when manual charging

Do not connect the manual or automatic charger to a long extension cable, it may get hot and cause a hazard situation.

Do not touch the copper contacts during the charging process in the charging station. You could suffer a small electric shock that could cause a slight burn.

Do not sprinkle water or oil on the robot or power charging cord. The exterior cleaning of the robot must be done with pressurized air or dry cloth. Contact with water or oil can cause electric shock or malfunction of the unit.

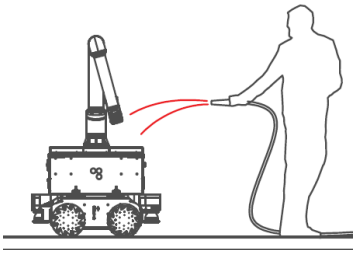


Figure 9 - Forbidden sprinkling

The robot should not be used as a support element or to transport people. You could damage the robot or injure yourself.

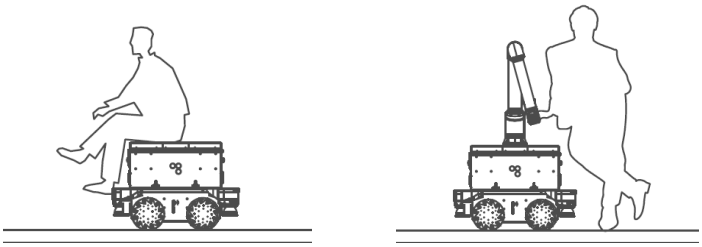


Figure 10 - Forbidden resting on the robot

The robot must not be lifted or pushed from any point. For lifting the vehicle, please use the 4 available points of grip (eyebolts), preferably with a crane. Always disassemble the robot arm before lifting it. Incorrect use could damage the robot or injure someone.

Figure 11 - How to lift the robot

The access to the robot electronics must always be done by disassembling in first place the covers without the robotic arm and, if necessary, disassembling the last cover too. Do not forget the robotic arm cover is the last element to remove. Please be careful when removing it. Incorrect use could damage the robot.

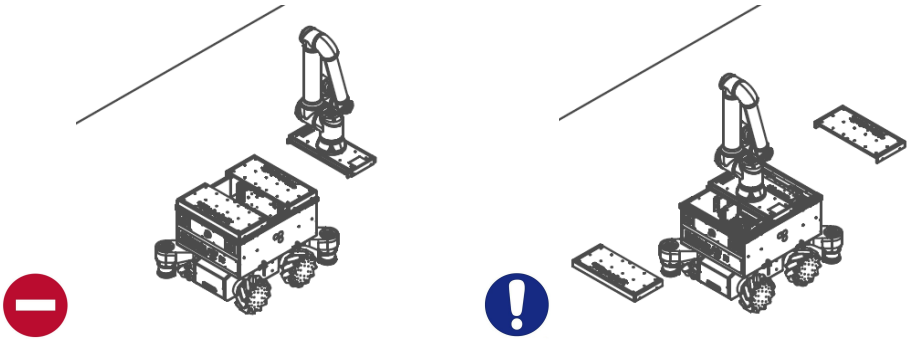


Figure 12 - Top cover disassembling order

If the RB-KAIROS+ runs out of battery in an unexpected situation, do not push it in order to move it or approach it to the battery charger. Otherwise, you could damage the robot motors.

Please approach the manual battery charger to the robot, wait a few minutes and drive the robot to the charging point.

Do not push the robot. Otherwise you could damage the robot motors.

Do not overload the robot nor the arm. Check their maximum payloads in their technical specification. Incorrect use could damage the robot or injure someone.

Do not drive the robot with mecanum wheels on wet surfaces. You could damage the robot.

Do not drive the robot outdoors when the weather is rainy. You could damage the robot.

Do not attempt to disassemble or modify the robot. You could damage the robot or injure yourself.

Do not modify installation parameters of the robot. You could damage the robot or injure someone.

Modifications or add-ons of hardware, and modifications of Robotnik software installation parameters could invalidate the safety and risk assessment.

Remember to update maps when the route changes to avoid new fixed obstacles. Not updating maps, you could damage the robot or injure yourself.

Always use the original charger and plug the power cord firmly into the wall outlet. Incomplete insertion in the wall outlet or the use of another charger could cause the plug to heat up, possibly causing a fire.

If there is evidence of battery malfunction, please follow the steps listed below. Use personal protective equipment, such as gloves, safety glasses and laboratory coat.

- If batteries show evidence of thermal runaway failure, be very cautious because the gasses may be flammable and toxic and failure modes can be hazardous.
- Disconnect the battery. Do not disassemble or break it.
- Remove the battery from the robot.
- Place the battery in a metal or other container away from combustibles.

In case of battery leakage entering into one's eye, rinse with water and look for medical advice immediately. Do not rub the eye.

The battery may get hot or ignite in case of electrical or mechanical abuse. In order to avoid these situations, follow the next instructions:

- Do not short-circuit the battery.
- Do not mechanically harm or disassemble the battery.
- Do not expose the battery to extreme temperatures, according to its manual.
- Do not allow the battery to get wet.
- Do not charge the battery outside the robot.

3.3. Specific risk situations

Specific risk situations related to laser detection are valid as long as the robot is equipped with certified safety lasers, and the lasers are not disabled.

STEPS UP

The robot is capable of climbing steps up to 30 mm in height. Any step of greater height is considered a barrier that can not be overcome by the robot and must be marked by a beacon located at the height of the laser vision. If obstacles are not marked with beacons, you could damage the robot or injure someone.

Figure 13 - Steps up marking

STEPS DOWN

The robot is capable of going steps down of 30 mm maximum height. Any higher step is considered a non-surmountable barrier by the robot. It must be marked by a beacon located at the height of the laser beam.If obstacles are not marked with beacons,**the robot may fall down through the steps and you could damage the robot or severely injure someone**.

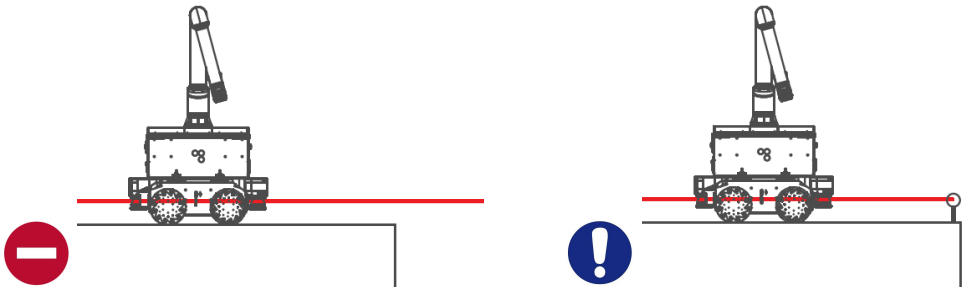


Figure 14 - Steps down marking

SLOPES

The robot is capable of climbing a maximum slope of 15%. Any greater slope can destabilize the robot, **cause its rollover and damage itself or severely injure someone.**

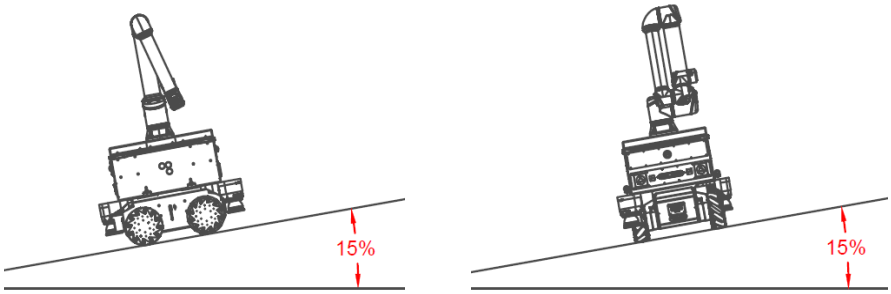


Figure 15 - Maximum slope

HOLES

When the robot is equipped with mecanum wheels, it is capable of driving through surfaces separated up to 20mm. Do not drive the robot on irregular surfaces or with separations over this distance. Otherwise you could damage the wheels and the robot.

WET OR SLIPPERY FLOOR

The robot is not prepared to drive on wet or slippery floor. When driving in slippery surfaces, the position and moving accuracy may decrease, and the brake distance will increase. These facts may lead to hazard situations. Do not drive the robot on slippery floor, otherwise you could damage the robot or injure someone.

NON-DETECTABLES OBSTACLES

When the platform is moving, the robot arm should be in a folded position inside the robot base footprint so as not to hit aerial obstacles. These obstacles could be:

- Door openings (in width and height).
- Obstacles on heights over the safety laser beam.
- Other components mounted on the robot.

Not avoiding these situations, you could damage the robot or injure someone.

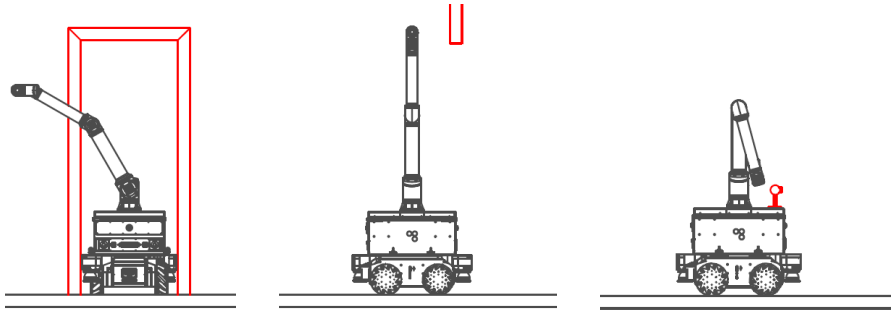


Figure 16 - Non-detectable obstacles marking

The safety sensors of the platform are not capable of detecting aerial obstacles if they are not correctly marked with beacons at the height of the laser beam. If obstacles are not marked with beacons, you could damage the robot or injure someone.

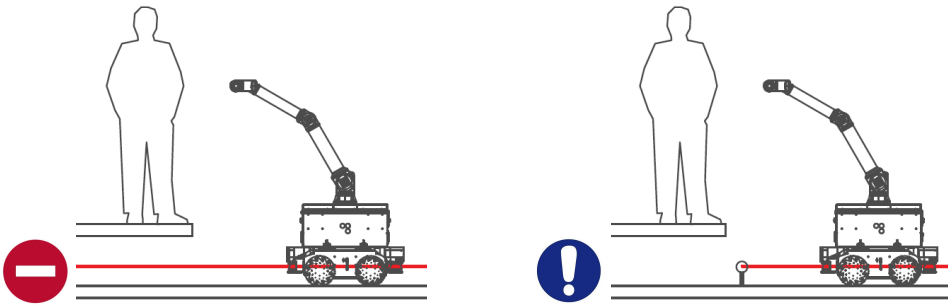


Figure 17 - Aerial obstacles marking

The safety sensors of the platform are not able to detect obstacles like transparent glass. The glasses must be marked with beacons at the height of the laser beam.If obstacles are not marked with beacons, you could damage the robot or injure someone.

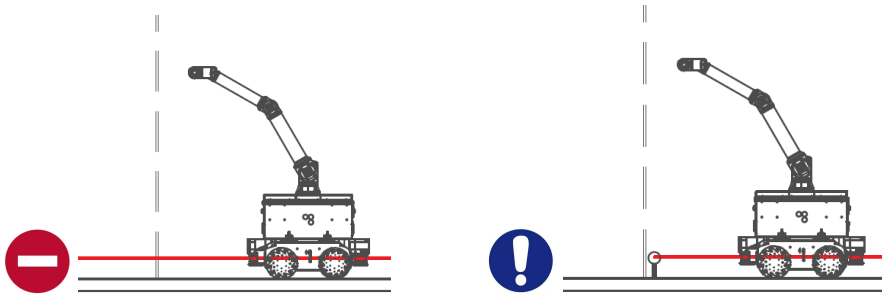


Figure 18 - Transparent glass marking

PERSON MOVING TOWARDS THE ROBOT

In automatic mode:

If someone moves towards the robot or steps into the robot path, it will be detected as an obstacle when inside of the safety zone and the robot will stop. Nevertheless, the safety zone does not take into account the robotic arm position. When driving the robot, please keep the arm inside the base footprint, otherwise you could injure someone.

For other working modes, please take in account the limitations and safety instructions of the corresponding mode. With Manual or Maintenance mode, the robot may not detect the presence of obstacles.Pay special attention to the robot and the environment. Otherwise you could injure someone.

MOVEMENT IN MANUAL MODE

Manual Mode should only be used by trained users, during commissioning or software development situations.The robot can be moved in manual mode, using the dualshock gamepad. In this mode, the user can choose to enable or disable the safety lasers. While moving in manual mode, you must pay attention to the robot and the environment. Otherwise, you could damage the robot or injure someone.

3.4. Safety zone

The default configuration of RB-KAIROS+ has two 2D lasers installed and located in two of its corners, which gives it a 360º vision range.

The vision range detects any obstacle located at a height of 170 mm from the ground. The RB-KAIROS+ is configured, as standard, to activate the emergency stop when an obstacle is detected.

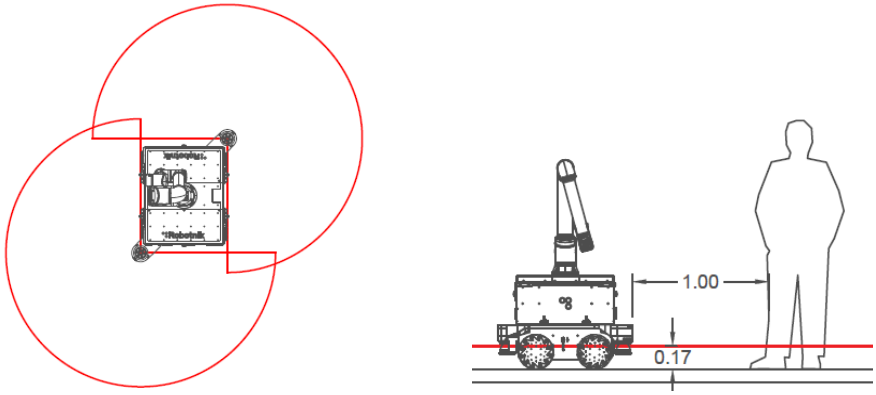


Figure 19 - Safety lasers detection field

The robot has a predefined safety zone on the horizontal plane, determined by safety lasers detection. When an obstacle or a person is detected inside the safety zone, the RB-KAIROS+ will stop until the area is free again. The size of the safety zone depends on the speed of the mobile base:

When the base is static or has a speed below 0.15 m/s, the safety zone is the one detailed in the scheme below:

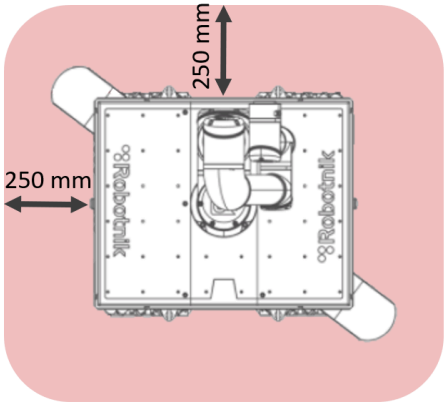


Figure 20 - Static safety zone

This area increases with the mobile base speed until reaching the maximum dimensions with a speed of 1m/s:

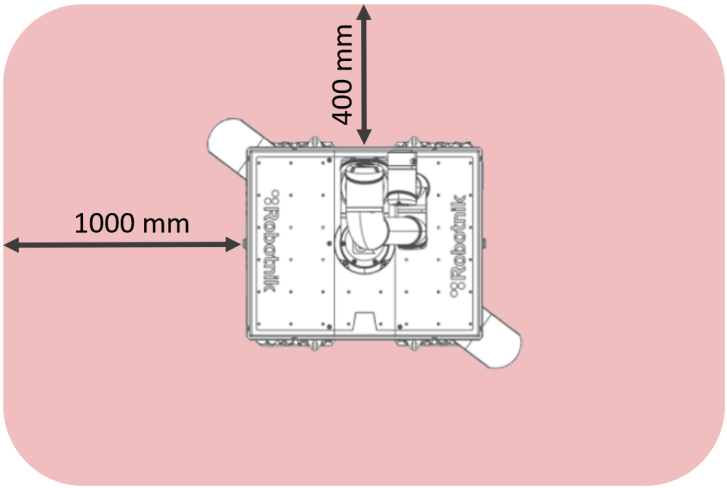


Figure 21 - Increased safety area

Once the maximum speed of 1.5m/s is reached, the platform stops for safety reasons.

This configuration can be modified according to the environment or application to be carried out, as long as it does not compromise the safety of the users.

WARNING

Modification of Robotnik installation parameters could invalidate the safety and risk assessment, and the guarantee of the robot. In case of modifying the safety zones it is necessary to perform a safety and risk assessment.

4. Quickstart

In order to unpack and mount the robot please follow in order the next instructions.

4.1. Unpacking the robot

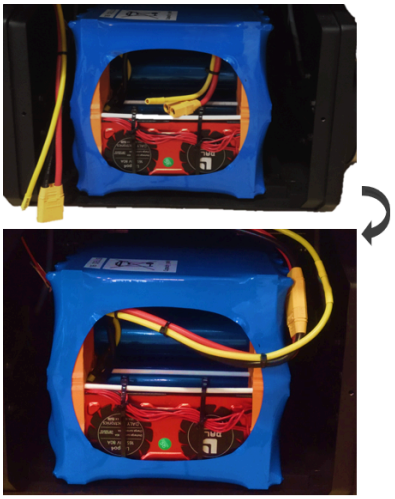
1.Disassemble the top cover
Unscrew the 4 screws and move away the top cover.

2.Disassemble the upper fixing bar
Unscrew the 2 screws and move it away.

3.Disassemble the front cover
You will identify the front part due to the presence of screws instead of nails. Unscrew the 10 screws of the front cover and move it away.

4.Disassemble the front fixing bar
Unscrew the 2 screws and move it away.

5.Disassemble the rear gate
Unscrew the 4 screws and take the gate off carefully. You will find the battery.



6.Connect the battery
Plug in the two connectors of the battery. Then keep them in a protected position and reassemble the rear gate.

7.Mount the exit ramp
Place one fixing bar at the foot of the box and use the top cover to mount an exit ramp for the robot. Please power on the robot and drive it outside of the box following the next steps on the manual.

4.2. Power on

In order to turn the robot on, you need to interact with the rear panel and follow the start-up sequence detailed below:

1. Turn the **Main power selector** to power on all the components of the robot.
2. For the unboxing process, check that the **Working mode key selector** is in *Manual mode*, and the **Laser key selector** is in *Laser mute* position.
3. Press the **Restart** button to give power to the motor drivers. Wait a few seconds until the robot is ready. The lights will turn green.

4.3. Pad teleoperation

Press the pad **Start** button to power on the dualshock gamepad.

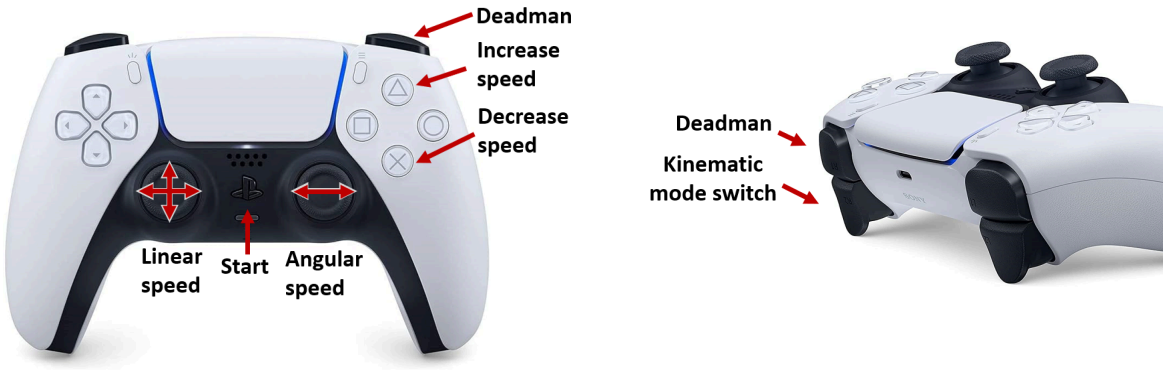


Figure 22 - Pad Operation

If the start-up sequence has been executed in the correct order, the pad will light up blue.Then, the robot will be ready to be teleoperated.

In order to move the robot, the **Deadman** button has to be pressed.

4.4. Driving the robot out of the box

With the robot released and powered on, and the exit ramp mounted, the robot is ready to be driven out of the box. The back side of the robot is facing the ramp, therefore you will have to drive it backwards.

Please press the Deadman button and move **slowly** the Linear speed button downwards.

Please be attentive to the environment of the robot. Otherwise you could injure someone or damage the robot.

Once the robot is out of the box and clear from fixed obstacles, the robot can be switched to Automatic mode. This is the working mode that will be used during the normal functioning of the robot. In order to do this:

1. Switch the **Working mode key selector** to *Automatic mode*.
2. Press the **Restart** button to give power to the motor drivers. Wait a few seconds until the robot is ready. The lights will turn green.

4.5. Power off

In order to turn off the robot, you need to interact with the rear panel and follow the sequence detailed below:

1. Press the **CPUs off** button. The robotic arm and base CPUs will turn off. Wait until the PC light indicators on the rear panel are off. The robot lights will turn blue.
2. Turn off the **Main power selector** to cut the power off. The lights will be switched off.

4.6. Arm mounting

Once the robot is turned off and out of the box, you can proceed to mount the robotic arm on the RB-KAIROS+ base. Please follow the instructions detailed below:

- If your RB-KAIROS+ has three top covers, and the arm will be mounted on the center one:

  1.Disassemble the rear top cover
  Unscrew the lateral screws of the rear top cover (the one that is closer to the rear panel) and move it away.

- If your RB-KAIROS+ has two top covers, and the arm will be mounted on the front one:

  1.1.Disassemble the rear top cover
  Unscrew the lateral screws of the rear top cover (the one that is closer to the rear panel) and move it away.

  1.2. Disassemble the front top cover
  Unscrew the lateral screws of the front top cover. Then lift it and place it at the center area of the RB-KAIROS+. This will allow you to have a better accessibility to the mounting area.

Figure 23 - Lateral top cover moved to the center

2.Place the pins
Place the two pins in the free holes of the top cover. Please notice that the pins have two different sides. Place the rounded side upwards.

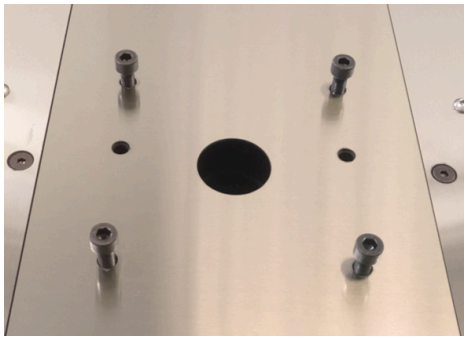


Figure 24 - UR locator pins

3.Unscrew the UR fixing screws
Unscrew the 4 UR fixing screws from the top cover.

4. Place the UR arm on the top cover
Approach the UR to the mounting position and introduce the cable into the center hole of the top cover. Align the UR base with the pins and screw holes, and place it on the top cover.
Please notice that the UR arm has a black circle on the base. For standard versions, this circle will be pointing to the left side of the RB-KAIROS+.

5.Screw the UR to the top cover
Tighten the four screws to 10Nm torque.

6. Connect the cable
Connect the UR cable inside the RB-KAIROS+ to communicate the arm with its control box.

7. Reassemble the top cover
Reassemble the top covers. Please pay attention not to damage the UR cable or any other component.

4.7. Storage

If the robot is going to remain inactive for a long period of time, please take in account the following recommendations:

Keep the robot stored indoors, protected from hazard conditions such as humidity, dust or extreme temperatures. Keep the robot between 0º and 50º, otherwise the battery and other components may be affected.

In order to protect the battery, fully charge the robot before storing it.

For further information about maintenance you can consult the RB-KAIROS+ maintenance manual and Universal Robots documentation.

5. Working modes

RB-KAIROS+ has three different working modes of the mobile base.
The base working modes are independent of the working modes of the robotic arm.

The base working mode affects safety elements and conditions.

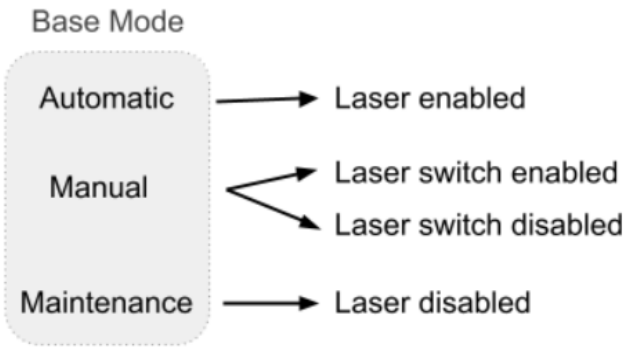


Figure 25 - Laser and base modes

5.1. Base in Automatic Mode

Base Automatic Mode is the working mode that will be used during the normal or autonomous functioning of your RB-KAIROS+ application. Therefore, this mode can be used by the daily operator user.

When the Automatic Mode is selected, the safety lasers are always enabled, independently of the laser key selector. Therefore, the different safety zones will be active, allowing RB-KAIROS+ to stop if any object appears in any protective zone.

For safety reasons, the base and the arm will only move under specific conditions.

5.1.1. Conditions for base movement

The safe movement of the robot will be enabled if the following conditions are met:

- All the emergency buttons are released and the RESTART button has been pressed.
- All the internal components are ready and powered.
- All the protective zones of the laser scanners are free.
- The arm is located or moving inside the robot's footprint.

5.1.2. Conditions for arm movement

The safe movement of the arm will be enabled if the following conditions are met:

- All the emergency buttons are released and the RESTART button has been pressed.
- All the safe zones of the laser scanners are free.
- The arm is powered on and initialized.
- The arm brakes are released.

There are two different types of “safe stop” when the arm is executing a program:

- When the emergency button is pressed while the arm program is running, the “emergency stop” is triggered, the program stops and it can’t be resumed.
- When the safe zones are occupied while the arm program is running, the “safeguard stop” is triggered and the program pauses. Once the safe zones are free again, the program is resumed automatically.

For a detailed explanation of the stop types please refer to the User Manual of the arm.

5.2. Base in Manual Mode

The Manual Mode should only be used by trained users, during commissioning or software development situations. With the base in Manual Mode, you can drive the RB-KAIROS+ using the dualshock gamepad, as shown previously in the chapter “ Pad Teleoperation”.

When the Manual Mode is selected, the laser key selector is activated. Depending on the chosen option, you can operate with the lasers and laser zones enabled, or disabled.

5.2.1. Safety lasers enabled

If the lasers are enabled, the conditions for base or arm movement are the same as in Automatic Mode.

5.2.2. Safety lasers disabled

If the lasers are disabled, they do not have influence neither in the conditions for base and arm movement, nor in the safety stop.

WARNING

The robot and the arm will move freely in any case if the key is in the “LASER MUTED” state as described in Rear panel section .

While moving RB-KAIROS+ in manual mode, you must pay attention to the robot and the environment. Otherwise, you could damage the robot or injure someone.

This is only recommended for experienced and trained users.

5.3. Base in Maintenance Mode

Base Maintenance Mode is the working mode recommended for RB-KAIROS+ maintenance.

When the Maintenance Mode is selected, the safety lasers are always disabled, independently of the laser key selector. Its behavior is the same as in Manual Mode with the lasers disabled, as described in the previous chapter.

WARNING

The robot and the arm will move freely in any case if the key is in the “MAINTENANCE” mode as described in rear panel section .

While moving RB-KAIROS+ in maintenance mode, you must pay attention to the robot and the environment. Otherwise, you could damage the robot or injure someone.

This is only recommended for maintenance users, with a medium knowledge level of electronics and mechanics.

6. Robot components

6.1. External components

6.1.1. Ports and power connectors

The external communication ports and power connectors are located in the rear panel of the RB-KAIROS+, as seen previously in “Main componentes”.

For more information, please refer to the annex “Pinout robot interfaces” of RB-KAIROS+ maintenance manual.

6.1.2. Motor wheels

The robot has 4 motor wheels with the same configuration. Each wheel is composed of a motor block and a detachable wheel. The motor block has a 500w 8 poles brushless motor with Hall Effect sensor and a reduction gear box, all of them held by an aluminum cover.
Brushless motors are used due to their longer service life and higher efficiency, in comparison to brushed motors.

The cables must be kept in good condition, and protected if the external cover is damaged.

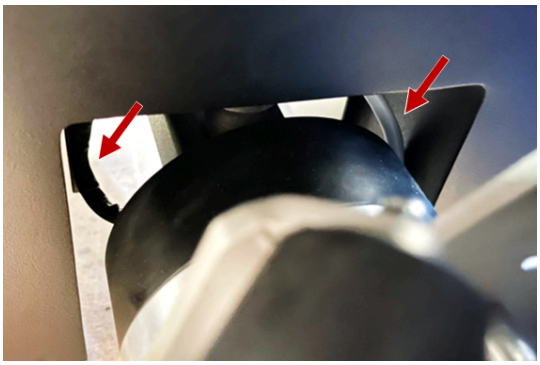


Figure 26 - Motor cables

There are two mecanum wheels (10 inch) options available for RB-KAIROS+:

- Standard mecanum wheel: payload 130 kg.
- Strong mecanum wheel: payload 250 kg.

Mecanum wheel is used in indoor environments. With these wheels, the platform achieves high precision of movement, provided that the floor meets the standards of cleanliness and flatness required for its use.

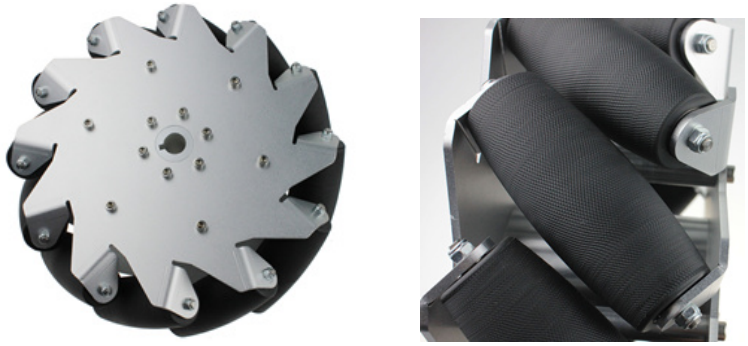


Figure 27 - Mecanum wheel

6.1.3. 3D camera sensor

3D camera sensor is an optional device for RB-KAIROS+. This sensor adds interesting features such as obstacle detection, robot location or reading QR markers. It allows, in addition, the option of configuring reactive navigation operations.

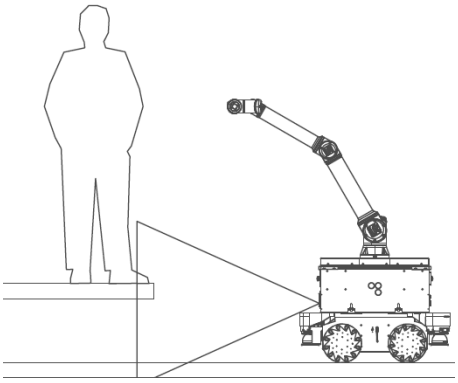


Figure 28 - 3D camera view

6.1.4. Acoustic and light signals

The RB-KAIROS+ safety pack includes acoustic and light signals to indicate the robot's actions when they affect the safety of the user or the robot itself.

LED LIGHT INDICATOR

RB-KAIROS+ light indicators give the following information:

	**No lights:** Robot switched off.
	Flashing white: Initializing.
	**Blue:** Robot not prepared to move. May be caused by an unknown ROS error, or robot PC not operative.
	**Green:** Robot ready to move.
	**Flashing Green:** Robot moving.
	**Green moving clockwise:** Robot turning right.
	**Green moving anticlockwise:** Robot turning left.
	**Flashing red:** Emergency stop. May be activated by E-stop button, safety laser zone or arm collision. May also be caused by motor failure, or safety lasers failure. Need to rearm the robot, or to clear the safety laser zone.
	**Fast Flashing orange:** Device anomaly. The robot may not navigate or may fail certain functionalities. May be activated due to a malfunction of the robotic arm or sensor, or miscommunication with these devices.
	**Flashing orange and green:** Low battery. The robot will continue with its normal behavior, as long as the battery lasts. Need to charge the robot.
	**Moving green and orange:** Charging.

ACOUSTIC INDICATORS

RB-KAIROS+ acoustic indicators give the following information:

- **Intermittent sound with a fixed frequency**: the base is moving

Sound configuration may be adapted depending on the task, but the robot is always delivered with the standard configuration described.

6.1.5. Robotic arm

RB - Kairos+ has always integrated a collaborative robotic arm from Universal Robots.
Depending on the reach and the payload of the arm, there are four different options that can be installed: UR3e, UR5e, UR10e, UR16e.

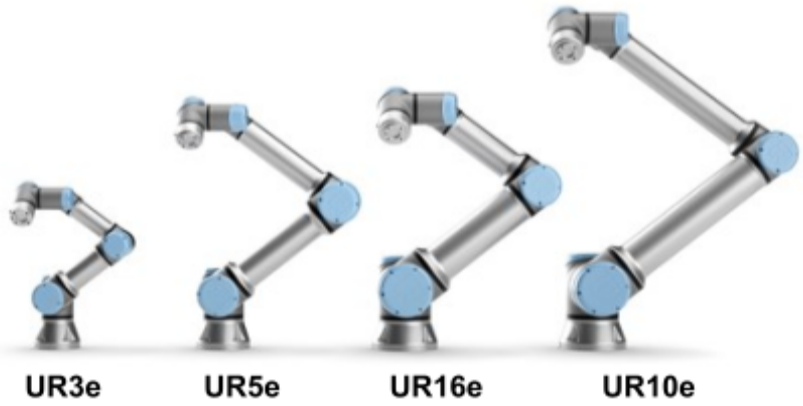


Figure 29 -UR arms

6.1.5.1. Reference axes and position

Depending on the type of arm chosen, its mounting position on the RB-KAIROS+ will change.By default, the robotic arm is installed at the center of the mobile base, in order to provide the system with high stability and low footprint (16e and 10e). Due to their shorter reach, 3e and 5e arms will be mounted on one side of the mobile base.

The different mounting configurations are shown in the following images:

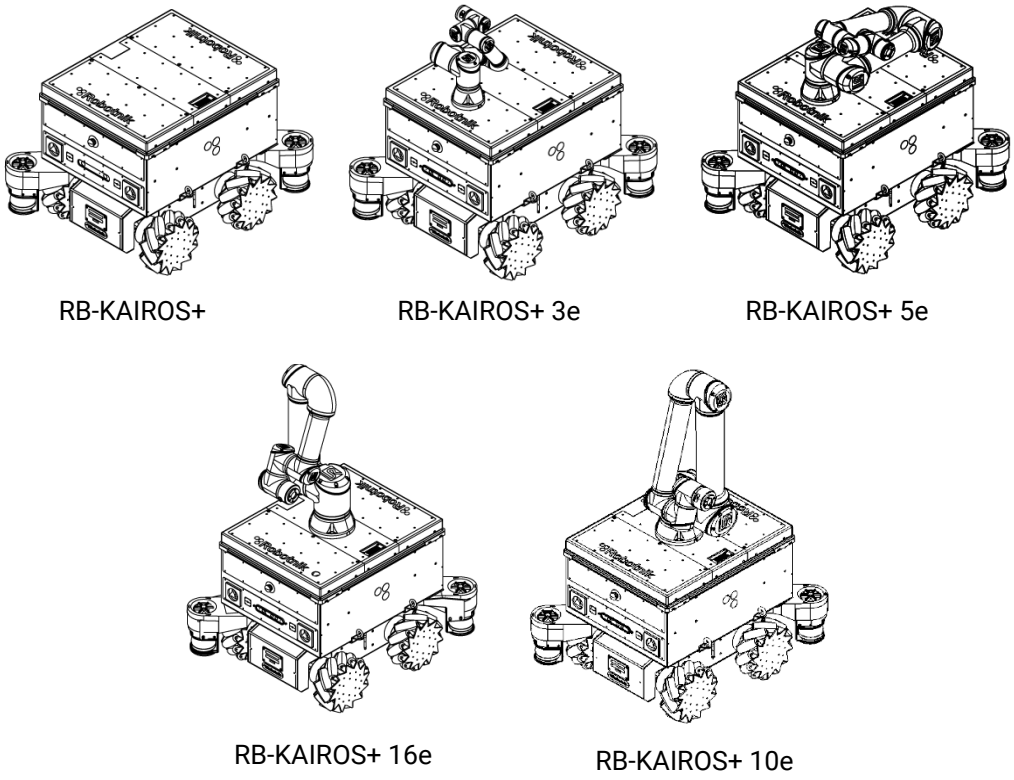


Figure 30 - UR mounting standard configurations

The reference axes of the arm base are located as shown at the following image:

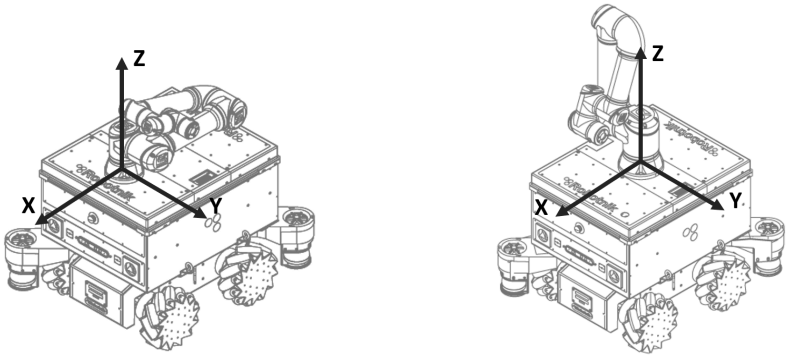


Figure 31 - UR axes configuration

If your robot has a different arm location, please consult UR documentation for further information.

6.1.5.2. Arm operating modes

The UR robotic arm has three different working modes: Remote Control mode, Automatic mode and Manual mode. The arm working modes are independent of the base working modes.

RB-KAIROS+ has by default the UR arm in Remote Control mode. This mode allows the automatic brake release and program execution when the robot is turned on.

Automatic and Manual mode enable the possibility of creating and testing programs with the arm, but not an automatic execution.

Please refer to the Universal Robots documentation for further information.

6.1.5.3. Safe operation zone

RB-KAIROS+ has defined by default an arm safety zone, configured in the arm installation. This zone is limited by five planes, corresponding to the limits of the mobile base.

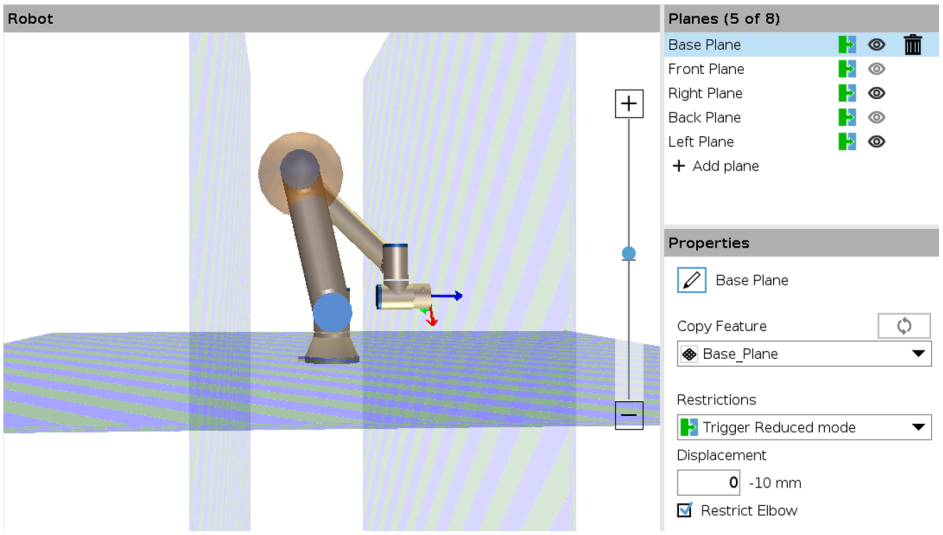


Figure 32 - Arm safety planes

When the flange and the elbow (3rd axis) of the arm are inside of the safety zone, maximum arm speed is available, and the mobile base is able to move.

When the flange and the elbow of the arm are outside of the safety zone, the arm gets in reduced mode. This means that the maximum arm speed available is the one defined as reduced speed in the arm installation. By default, when the base is in automatic mode, it will not move if the arm is in reduced mode.

WARNING

The safe operation zone and the reduced mode of the robotic arm, does not avoid arm movement. You must pay attention to the robot and the environment. Otherwise, even with a reduced arm speed, you could damage the robot or injure someone.

6.1.6. Other devices

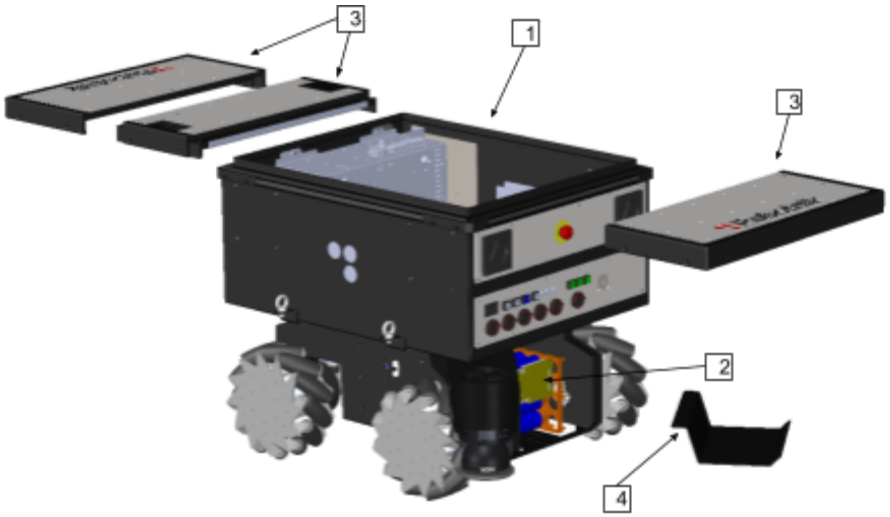
RB-KAIROS+ is prepared to integrate other sensors not mentioned, depending on the solution needed.
The top covers of the RB-KAIROS+ are prepared with several metric 6 holes on their upper face, in order to receive the installation of any screwed component. A cable entry system is also included, for allowing the pass of the wiring inside the robot structure.

WARNING

The installation of any device performed out of Robotnik facilities could invalidate the safety assessment and the warranty.

6.2. Internal components

Any element inside of the robot chassis is considered as an internal component. The next figure shows the main internal components of the default robot:

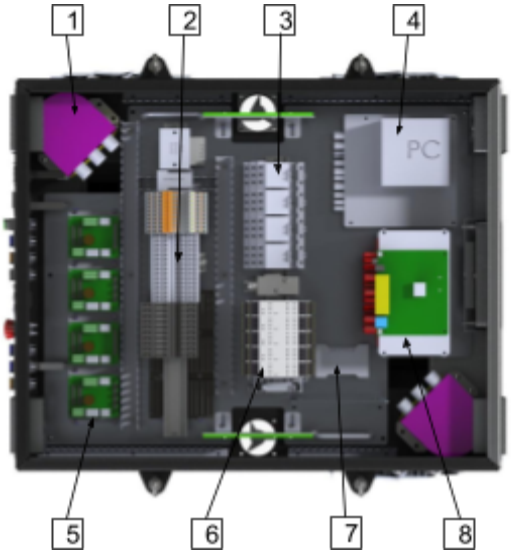


1. Electronic board
2. Battery
3. Top covers
4. Back gate

Figure 33 - Internal components of RB-KAIROS+

6.2.1. Electronic board

The next figure shows the components of the default RB-KAIROS+ electronic board:



1. 2D Laser
2. Fuses and terminals
3. Motor relays
4. Platform CPU
5. Motor drivers
6. Safety modules
7. IMU
8. Motherboard arm and router

Figure 34 - Components of RB-KAIROS+ electronic board

Among all these components, the following parts should be highlighted:

CPU. There are two CPUs in the RB-KAIROS+: the mobile base CPU and the UR Motherboard Arm. The first one executes ROS and the second one executes UR driver software (PolyScope).

Motor driver. There are four drivers which are programmed with specific settings to control each motor wheel of the platform.

IMU. The Inertial Measurement Unit (IMU) is an electronic device that measures and reports a robot's specific force, angular rate and magnetic field.

There are different models of these components that can be installed in RB-KAIROS+. For further information about them, please check their specific documentation or contact Robotnik support department.

6.2.2. Battery

The robot receives the power supply from a LiFePO4 battery pack. It is composed of 3.2V LiFePO4 cells and a protection circuit module. With this battery technology the robot is able to operate between 3 and 10 hours, depending on the battery capacity chosen and the robot movements. The default battery pack is 30Ah@48VDC, if you have a different battery and want further information, please contact Robotnik support department.

Figure 35 - 30Ah@48VDC Battery pack

The robot circuit is powered when the general switch is ON. The control DC/DC converter, which supplies power to the different devices of control, is powered at the same time.

In case of battery malfunction, please consult the General safety instructions chapter, and the RB-KAIROS+ maintenance manual.

6.3. Accessories

6.3.1. Dualshock gamepad

The dualshock gamepad is a default robot item and it is used for the manual movements of the robot. Its receiver is located inside the robot and connected to one USB port of the computer.

Figure 36 - Dualshock gamepad

Please keep in mind to keep charged the dualshock gamepad so that is ready for its next use.

NOTES

- The " **Kinematic Mode Switch**" is only valid when the robot has installed the mecanum wheels and the controller is configured to work in omni-directional mode.
- If the **Deadman** button is NOT pressed, the robot will NOT receive commands from the remote controller.
- Sometimes it is necessary to restart the pad to link it again with the robot controller. To restart it, press the **Start** button until the light shuts down and afterwards proceed with the startup sequence.

For further information about the DualShock, please consult its documentation or contact the Robotnik support department.

6.3.2. Manual charger

The manual charger is a default robot device and it is used for charging the robot manually. Depending on the robot battery, the manual charger has different charging power.

Figure 37 - Manual charger

The battery connector is located at the rear panel of the robot. It is a direct connection to the battery, so the general ON/OFF switch doesn't affect the charging.

It is possible to charge the robot and keep working at the same time. Nevertheless this situation is not optimal for the battery service life. That is why this option is only recommended during a short period of time for specific situations, such as commissioning or programming processes.

The robot platform should not be moved when the robot is connected to the manual charger. Otherwise, you could damage the robot or the charger.

The standard manual charger (Mean Well HEP) has a led indicator with the following signals:

- Green: Floating state. Robot battery is completely charged or the charger is not connected.
- Red: Charging.

For further information about the different charger models, please contact the Robotnik support department.

7. Control

RB-KAIROS+ and its software applications can be controlled and monitored through different systems:

- RB-KAIROS+ is equipped with the Robotnik Human Machine Interface (HMI), an internal web application running locally on the robot. For further information and instructions about the HMI, please consult the Control Interface manual.

- The UR arm can be controlled with the Polyscope interface. For further information about it, please refer to the UR official documentation. You can find all related manuals in UR official webpage (https://www.universal-robots.com/download).

- For advanced developers, RB-KAIROS+ can be controlled through ROS programming. For further information, please refer to the Developer manual.

In the following chapters it is described how to connect to the robot and software credentials, regardless of the software used.

7.1. Connecting to the robot

The standard RB-KAIROS+ is equipped with a router and its own Wi-Fi network, identified by its serial number. If your robot has a Wi-Fi network, it will allow you to connect to the robot's local network. Your static IP must be in the 192.168.0.X range and can not be one that is already being used. Take as an example the 192.168.0.100.

If your robot does not have a router, you will need to connect to the robot via Ethernet using the LAN port in the rear panel. You will have to configure your IP to be static in order to communicate with the robot. The most convenient way to work with the robot is to connect via cable for the first time and configure its Wi-Fi interface to connect to a network that you have available in your facilities, as explained below in the Wi-Fi setup chapter.

7.1.1. Wi-Fi setup

If the robot is not standard and does not have a wi-fi router, it may have a wi-fi interface. In that case, the wi-fi can be configured following the next steps:

1. Connect to the robot through SSH or VNC following the steps described previously.
2. Execute nmtui command:

```
nmtui
```

3. Select “Activate a connection”:

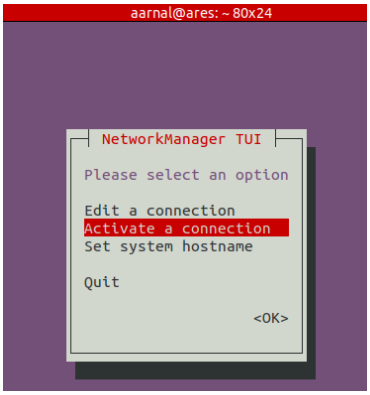


Figure 38 - Wifi set-up 1

4. Select the desired network connection:

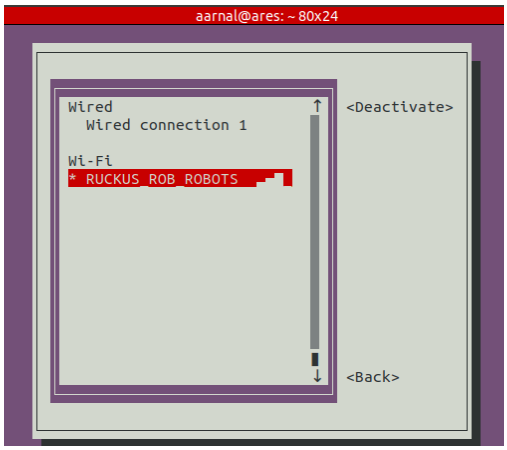


Figure 39 - Wifi set-up 2

Keep in mind that:

- All credentials can be found in Users and passwords chapter.
- The IP addresses of the devices may change depending on your specific configuration. All standard IP addresses are listed in Network chapter.

7.1.2. Mobile base

SSH

To communicate through ssh and opening a remote terminal from the robot’s CPU type the following command:

```
ssh robot@192.168.0.200 -Y
```

VNC
To connect to the robot through VNC you can use a VNC client (i.e. Remmina on Ubuntu)

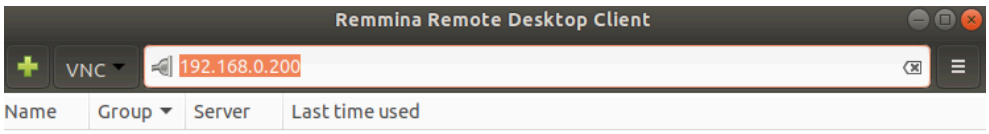


Figure 40 - VNC client

7.1.3. UR Arm

To connect to the arm, you need to be connected to the robot's local network, for example:

- Connected to Wi-Fi* of the robot.
- Connected by Ethernet to the rear panel of the robot.
- Connected to the robot's CPU.

*This is only applicable in case your robot has a router.

SSH
To communicate through ssh and opening a remote terminal from the arm's CPU type the following command:

```
ssh root@192.168.0.210 -Y
```

VNC
The UR arm has installed a VNC server to enable remote access to the UR Polyscope). By default the vnc server should start on boot but if it does not start, you can force the server from the base PC:

1. Make sure the arm is powered on.
2. Start the VNC server on the UR PC.

   ```
   ssh root@192.168.0.210 bash -c './start_vnc.sh'
   ```

   NOTE: The credentials for the PC of the arm are provided in Users and passwords chapter

3. Once the command, it will return the port where the VNC server is located.
4. Connect to the VNC server using the IP of the arm and the returned port.

For further information, please refer to the Developer manual.

7.2. Network

In the following diagram there are the most common network connected devices with their IP addresses. This information only applies to the components present in your robot.

Device	IP address
Mobile platform CPU	192.168.0.200
UR Arm	192.168.0.210
Router	192.168.0.1
Front camera	192.168.0.185
Rear camera	192.168.0.186
Front Laser	192.168.0.10
Rear Laser	192.168.0.11

Accessory device	IP address
OnRobot control box	192.168.0.213

7.3. Devices

The devices connected to the robot have two ways of connection: via network or via USB. If they are connected through the robot network, they are configured with a static IP to have them identified (as shown previously in Network chapter). If connected by USB, udev rules are usually defined to identify devices with names. The names and IPs of the devices are defined as environment variables.

7.4. Users and passwords

In this section you can find the user and password configurations of the most common devices. This information only applies to the components present in your robot.

Mobile platform CPU	
User	robot
Password	R0b0tn1K

UR Arm	
User	root
Password	easybot
Safety and Mode password	ur

Router[1]	
User	admin
Password	R0b0tn1K
SSID	SXLSK-YYMMDDAA_2G (or 5G)[2]
Password	R0b0tn1K

1. This applies only if your robot has a Wi-Fi router.
2. The SSID will vary depending on the serial number of your product.

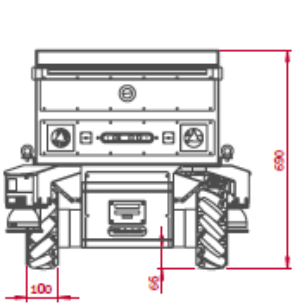
ACCESSORY DEVICES

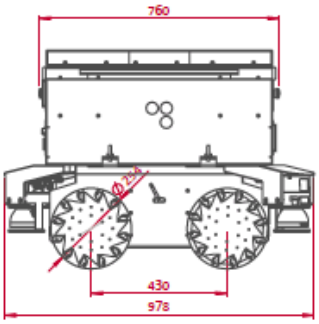
OnRobot control box	
User	admin
Password	R0b0tn1K

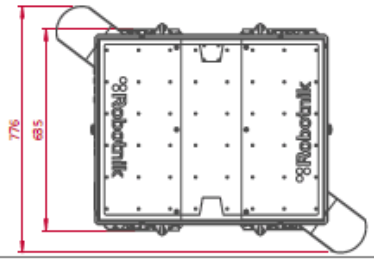
8. Technical specifications

8.1. RB-KAIROS+ base

RB-KAIROS+ Robotnik







TECHNICAL SPECIFICATIONS

MECHANICAL

Dimensions	978 x 776 x 690 mm
Weight	115 kg
Payload	Up to 250 kg
Speed	2 m / s
Environment	Indoor
Enclosure class	IP52
Autonomy	Up to 12 h
Batteries	LiFePO4 15Ah@48VDC or 30Ah@48VDC
Traction motors	4 x 500 W Brushless servomotors with safety brake
Temperatura range	0° to +50°C
Max. slope	15 %

CONTROL

Controller	Open architecture ROS Integrated PC with Linux
Communication	WiFi 02.11a / b / g / n / ac Bluetooth 5.1 (5G/4G optional)
Connectivity	USB, RJ45, power supplies 12, 24 VDC and battery
	Ready for plug and play integration of Universal Robots™ arms (OEM DC Version)

ROS www.robotnik.eu

8.2. RB-KAIROS+3e

Robotnik

RB-KAIROS 3e

COMPLETELY INTEGRATED
COLLABORATIVE MOBILE MANIPULATOR (CMM), FOR
INDUSTRIAL TASKS

PICK & PLACE
METROLOGY
PART FEEDING
INDUSTRIAL APPLICATIONS
R & D

TECHNICAL SPECIFICATIONS

MECHANICAL

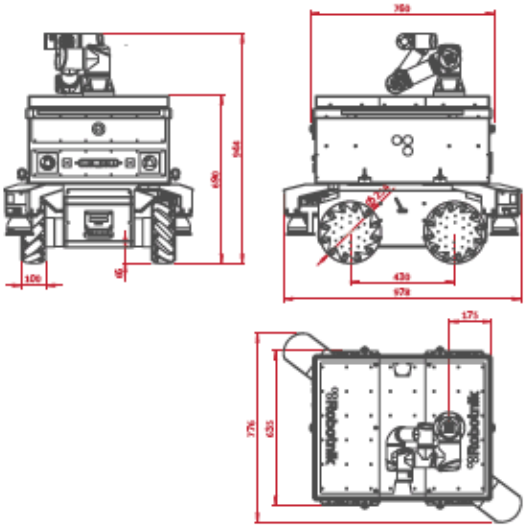
Dimensions	**978 x 776 x 944 mm**
Weight	**115 kg + 11,2 kg**
Payload	**Up to 250 kg**
Speed	**1,5 m / s**
Environment	**Indoor**
Enclosure class	**IP52**
Autonomy	**Up to 12 h**
Batteries	**LiFePO$_4$ 15Ah@48VDC or 30Ah@48VDC**
Traction motors	**4 x 500 W brushless servomotors wtih safety brake**
Temperatura range	**0° to + 50°C**
Max. slope	**15 %**

ROS

www.robotnik.eu

RB-KAIROS 3e

Robotnik

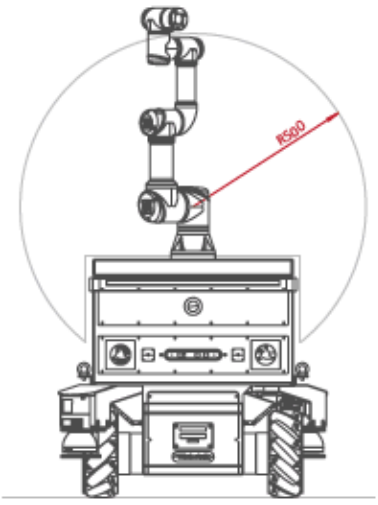


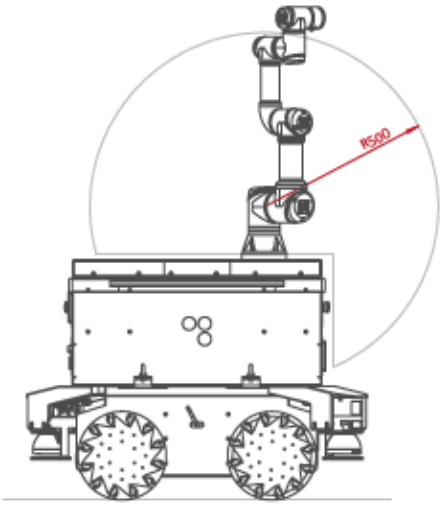
CONTROL

Controller	**Open architecture ROS Integrated CPU with Linux**
Communication	**WiFi 02.11a / b / g / n / ac Bluetooth 5.1 (5G/4G optional)**
Connectivity	**USB, RJ45, power supplies 12, 24 VDC and batteries**
	Ready for plug and play integration of Universal Robots™ arms (OEM DC Version)

UR3e

Payload	**3 kg**
Reach	**500 mm**
	Ready for integration of compatible accessories arm available in ROS Components store





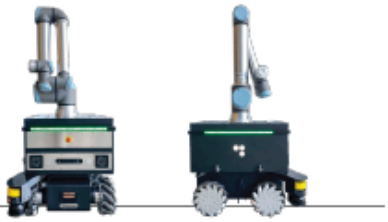
ROS

www.robotnik.eu

8.3. RB-KAIROS+5e

Robotnik

RB-KAIROS 5e



COMPLETELY INTEGRATED
COLLABORATIVE MOBILE MANIPULATOR (CMM), FOR
INDUSTRIAL TASKS

PICK & PLACE
METROLOGY
PART FEEDING
INDUSTRIAL APPLICATIONS
R & D

TECHNICAL SPECIFICATIONS

MECHANICAL

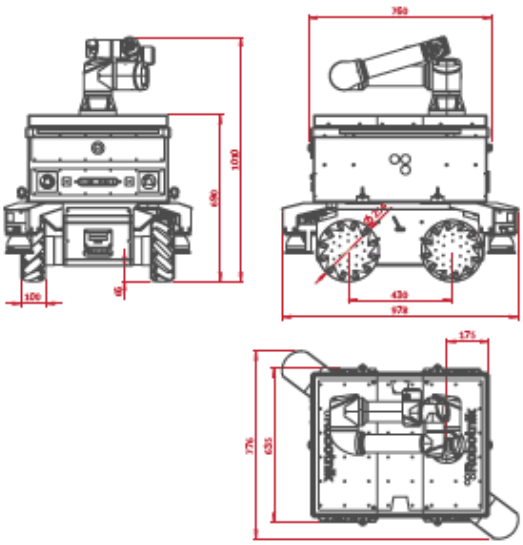
Dimensions	**978 x 776 x 1.010 mm**
Weight	**115 kg + 20,6 kg**
Payload	**Up to 250 kg**
Speed	**1,5 m / s**
Environment	**Indoor**
Enclosure class	**IP52**
Autonomy	**Up to 12 h**
Batteries	**$LiFePO_4$ 15Ah@48VDC or 30Ah@48VDC**
Traction motors	**4 x 500 W brushless servomotors wtih safety brake**
Temperatura range	**0° to + 50°C**
Max. slope	**15 %**

ROS

www.robotnik.eu

RB-KAIROS 5e

Robotnik

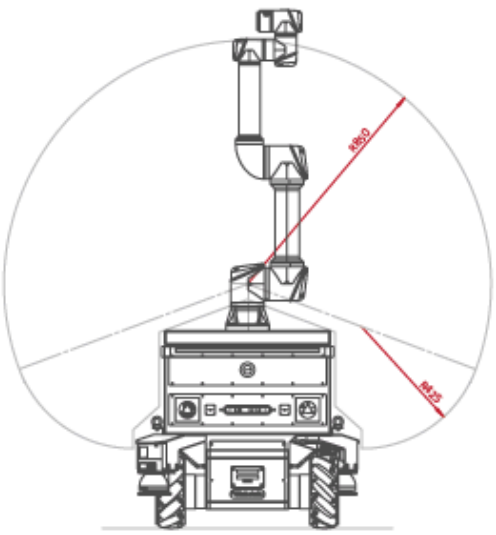


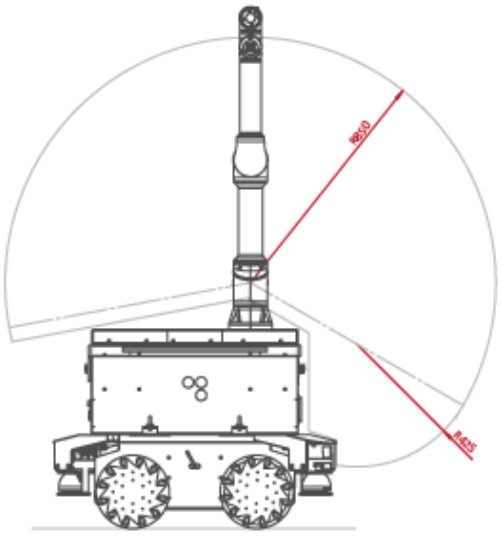
CONTROL

Controller	**Open architecture ROS Integrated CPU with Linux**
Communication	**WiFi 02.11a / b / g / n / ac Bluetooth 5.1 (5G/4G optional)**
Connectivity	**USB, RJ45, power supplies 12, 24 VDC and batteries**
	Ready for plug and play integration of Universal Robots™ arms (OEM DC Version)

UR5e

Payload	**5 kg**
Reach	**850 mm**
	Ready for integration of compatible accessories arm available in ROS Components store





ROS

www.robotnik.eu

8.4. RB-KAIROS+16e

Robotnik

RB-KAIROS+ 16e

COMPLETELY INTEGRATED
COLLABORATIVE MOBILE MANIPULATOR (CMM), FOR
INDUSTRIAL TASKS

PICK & PLACE
METROLOGY
PART FEEDING
INDUSTRIAL APPLICATIONS
R & D

TECHNICAL SPECIFICATIONS

MECHANICAL

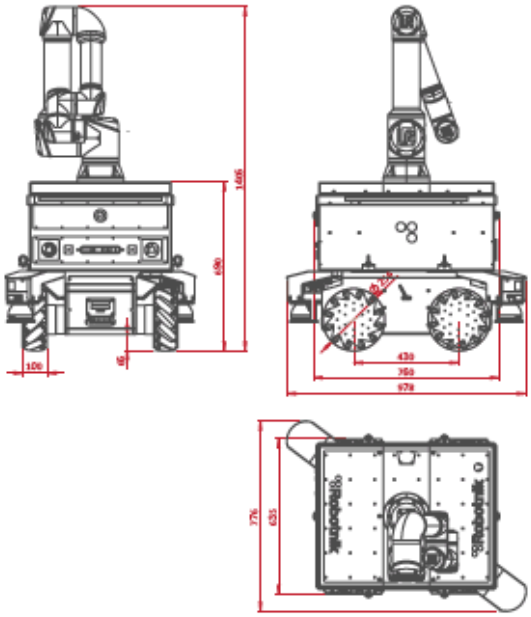
Dimensions	**978 x 776 x 1.406 mm**
Weight	**115 kg + 33,1 kg**
Payload	**Up to 250 kg**
Speed	**1,5 m / s**
Environment	**Indoor**
Enclosure class	**IP52**
Autonomy	**Up to 12 h**
Batteries	**LiFePO4 15Ah@48VDC or 30Ah@48VDC**
Traction motors	**4 x 500 W brushless servomotors wtih safety brake**
Temperatura range	**0° to + 50°C**
Max. slope	**15 %**

ROS

www.robotnik.eu

RB-KAIROS 16e

Robotnik

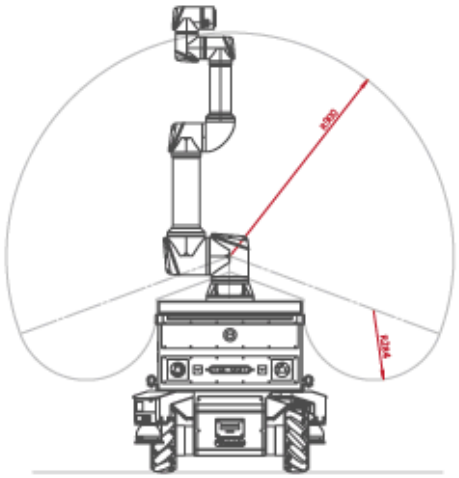


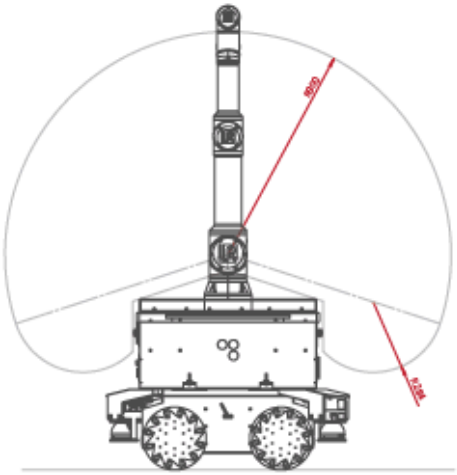
CONTROL

Controller	**Open architecture ROS Integrated CPU with Linux**
Communication	**WiFi 02.11a / b / g / n / ac Bluetooth 5.1 (5G/4G optional)**
Connectivity	**USB, RJ45, power supplies 12, 24 VDC and batteries**
	Ready for plug and play integration of Universal Robots™ arms (OEM DC Version)

URIGE

Payload	**16 kg**
Reach	**900 mm**
	Ready for integration of compatible accessories arm available in ROS Components store





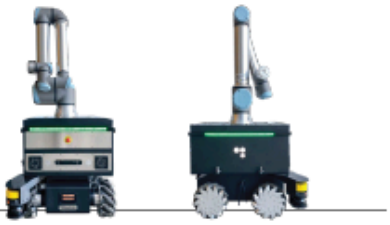
ROS

www.robotnik.eu

8.5. RB-KAIROS+10e

Robotnik

RB-KAIROS+ 10e



COMPLETELY INTEGRATED
COLLABORATIVE MOBILE MANIPULATOR (CMM), FOR
INDUSTRIAL TASKS

PICK & PLACE
METROLOGY
PART FEEDING
INDUSTRIAL APPLICATIONS
R & D

TECHNICAL SPECIFICATIONS

MECHANICAL

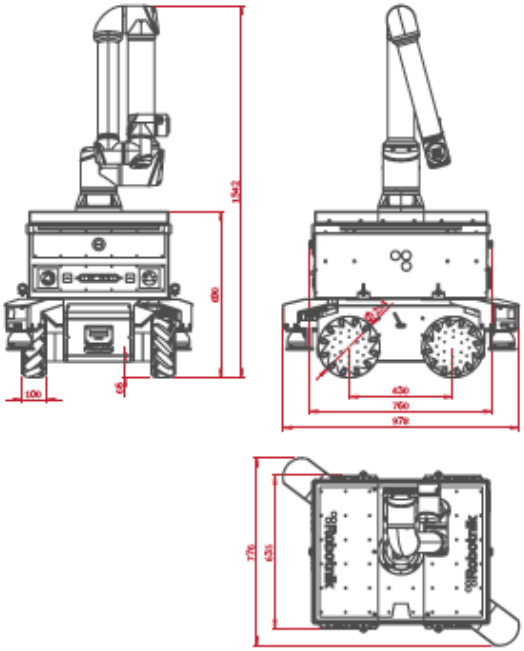
Dimensions	**978 x 776 x 1.542 mm**
Weight	**115 kg + 33,5 kg**
Payload	**Up to 250 kg**
Speed	**1,5 m / s**
Environment	**Indoor**
Enclosure class	**IP52**
Autonomy	**Up to 12 h**
Batteries	**LiFePO4 15Ah@48VDC or 30Ah@48VDC**
Traction motors	**4 x 500 W brushless servomotors wtih safety brake**
Temperatura range	**0° to + 50°C**
Max. slope	**15 %**

ROS

www.robotnik.eu

RB-KAIROS 10e

Robotnik

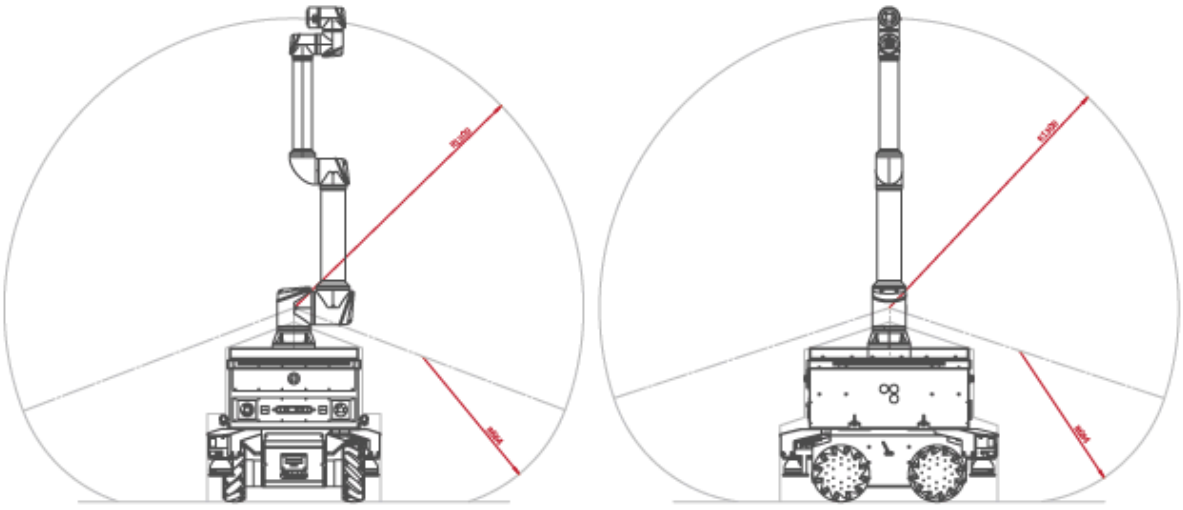


CONTROL

Controller	**Open architecture ROS Integrated CPU with Linux**
Communication	**WiFi 02.11a / b / g / n / ac Bluetooth 5.1 (5G/4G optional)**
Connectivity	**USB, RJ45, power supplies 12, 24 VDC and batteries**
	Ready for plug and play integration of Universal Robots™ arms (OEM DC Version)

URIOE

Payload	**12,5 kg**
Reach	**1.300 mm**
	Ready for integration of compatible accessories arm available in ROS Components store



ROS

www.robotnik.eu

9. Related standards and documentation

RB-KAIROS+ complies with the relevant provisions of the following EU directives or regulations.

- **2006/42/EC.** Directive 2006/42/EC of European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC.

Reference to the harmonized standards used, as referred to in Article 7.

- **EN 60204-1:2006/A1:2019**. Safety of machinery - Electrical equipment of machines - Part 1: General requirements.
- **EN ISO 12100:2010-11**. Safety of machinery – General principles for design – Risk assessment and risk reduction (ISO 12100:2010).
- **EN ISO 13849-1:2015**. Safety of machinery – Safety related parts of control systems – Part 1: General principles for design (ISO 13849-1:2015).
- **EN ISO 13849-2:2012**. Safety of machinery – Safety related parts of control systems – Part 1: Validation (ISO 13849-2:2012).
- **EN ISO 13850:2015**. Safety of machinery – Emergency stop function – Principles for design (ISO 13850:2015).
- **EN 60204-1:2006-6**. Safety of machinery – Electrical equipment of machines – Part 1: General requirements.
- **EN 1175-1:1998+A1:2010**. Safety of industrial trucks – Electrical requirements – Part 1: General requirements for battery powered trucks.

Reference of the other technical standards and specifications used:

- **ISO 3691-4**. Industrial trucks – Safety requirements and verification – Part 4: Driverless industrial trucks and their systems.

10. Guarantee

The guarantee of the products offered will be one year from the date of delivery of products, in case of subsystems provided by thirty parts i.e. the Arms, the period provided by the manufacturer. It includes replacement parts and delivery of the parts, it does not include the mounting of the damaged parts at client facilities. Robotnik will provide technical support for hardware and software via e-mail, phone and previously scheduled Skype telemeeting (in English or Spanish) during Robotnik working hours, for the warranty period. We provide free access to the software updates to our clients even after the guarantee period.