qb robotics

General Manual

PLEASE READ CAREFULLY THESE INSTRUCTIONS BEFORE USE. DO NOT DISCARD: KEEP FOR FUTURE REFERENCE.

Dear customer,

Thank you for purchasing our product.

The present document provides information at best of our knowledge at the time of publication. This document could present differences from the product and it is subject to changes without notice: the latest version is available on our webpage www.qbrobotics.com.

qbrobotics s.r.l. does not assume any responsibility for errors or omissions.

In no case qbrobotics s.r.l. will be responsible for any loss, problems or damages to persons or property arising from the use of this document.

Table of Contents

1 Preface

1.1 Using this document

The documentation must always be complete and in a perfectly readable state.

Keep the document accessible to the operating and, if necessary, maintenance personnel at all times.

Pass the document to any subsequent owner or user of the product.

1.2 Symbols and conventions

STYLE	DESCRIPTION
❗ **CRITICAL:** …	Identifies information about practice or circumstances that can lead to critical damages on the device and to personal injury. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
⚠ **WARNING:** …	Identifies information about practice or circumstances that can lead to damages on the device and to personal injury. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
ⓘ **NOTE:** …	Identifies remarkable information and additional notes.

STYLE	DESCRIPTION
✓ **TIPS:** …	Identifies tips and highlights.
	Identifies the required tool to be used during the described assembly phase.
`Monospaced text`	Identifies file paths, file names and software functions.

1.3 Kit content

The kit consists of:

- N.1 qb SoftHand2 Research 24 V;
- N.1 ISO 9409-1-50-4-M6 flange adapter;
- N.1 3m main cable;
- N.1 USB to RS485 converter with extension cord;
- N.1 USB flash drive:
  - GUI for Linux and Windows;
  - Manual;
  - Datasheet;
- N.1 cylindrical pin EN ISO 8734 A d6x14 h6;
- N.4 metrical screws EN ISO 4762 M6x10;
- N.8 metrical screws EN ISO 10642 M3x8;
- N.1 2 mm Allen hex key;
- N.1 5 mm Allen hex key.

Optionals:

- 24 VDC power supply unit;
- ISO 9409-1-40-4-M6 flange adapter;
- ISO 9409-1-31.5-4-M5 flange adapter;
- Kinova Gen3 flange adapter.

2 Safety

2.1 Intended use

The product design is intended for grasping and manipulating objects in the weight range from 1 to 1700 g.

Fragile, sharp or sharp-edged objects shall not be grasped.

Objects having the ratio between the biggest and the smallest dimensions ≥8 mm and the main dimension greater than 300 mm shall not be grasped.

The product, as its name suggests, is intended for research and educational use.

The product is intended for installation on robotic arms: the safety features are established only for use as described in this document.

The safety of the product cannot be guaranteed in case of inappropriate use. One, single, inappropriate use can result in a permanent damage to the safety of the product.

2.2 Safety instructions

WARNING:

- Check that all the content is intact after removing it from the packaging.
- The device can be used only by specially trained staff.
- Disconnect the power supply before installation, cleaning or maintenance operations.
- Make sure that no residual energy remains in the system.
- Always operate the product within the specifications defined.
- Keep away from children and pets. Always set off or unplug when not in use.
- Never use aerosol products, petroleum based lubricants or other flammable products on or near the end-effectors.
- Do not use any damaged power cable, plug, or loose outlet. It may cause damages to the product or injury to people.
- Do not touch electrical components to avoid damages due to electrostatic charges.
- Make sure the end-effector is properly and securely bolted in place and cabled.
- Do not use if damaged or defective. Do not disassemble.
- Do not insert any objects between moving parts of the fingers.
- Keep head and face outside the reach of the end-effector.
- Do not wear loose clothing or jewelry when working with the end-effector.
- Disrespect of these precautions can affect safety of the device.

2.3 EC Directives on product safety

- The following EC directives on product safety must be observed.
- If the product is being used outside the UE, international, national and regional directives must be also observed.

2.3.1 Machinery Directive (2006/42/EC)

Because of their small size, no serious threats to life or physical condition can normally be expected from electric miniature drivers. Therefore, the Machinery Directive does not apply to our products. The products described here are not "incomplete machines", so installation instructions are not normally issued by qbrobotics.

2.3.2 Low Voltage Directive (2014/35/EU)

The Low Voltage Directive applies for all electrical equipment with a nominal voltage of 75 to 1500 V DC and 50 to 1000 V AC. The products described in this device manual do not fall within the scope of this directive, since they are intended for lower voltages.

2.4 Environmental conditions

Wrong environmental and operating conditions can lead to injuries, product damages and/or significant reduction to the product's life.

⚠

WARNING:

Any use or application deviating from intended use is deemed to be impermissible misuse. This includes, but is not limited to:

- Use before performing a risk assessment;
- Use outside the permissible operational conditions and specifications;
- Use in not low-dust environment;
- Use in places with high temperature or humidity;
- Use in wet places;
- Use in potentially explosive atmospheres;
- Use in medical and life critical applications;
- Use close to a human's head, face and eye area;
- Use as a climbing aid;
- Use in outdoor applications.

2.5 Environmental safety

The qb SoftHand2 Research must be disposed of in accordance with the applicable national laws, regulations and standards.

All the components of this product have been chosen in accordance with the EU RoHS directive 2011/65/EU: they are produced with restricted use of hazardous substances to protect the environment.

Observe national registration requirements for importers according to EU WEEE Directive 2012/19/EU.

3 Technical data

3.1 Mechanical dimensions

The figure below shows the overall dimensions of the qb SoftHand2 research, valid for both left and right configurations; distances and tolerances in the drawing are noted in millimeters `[mm]` and degrees `[°]`.

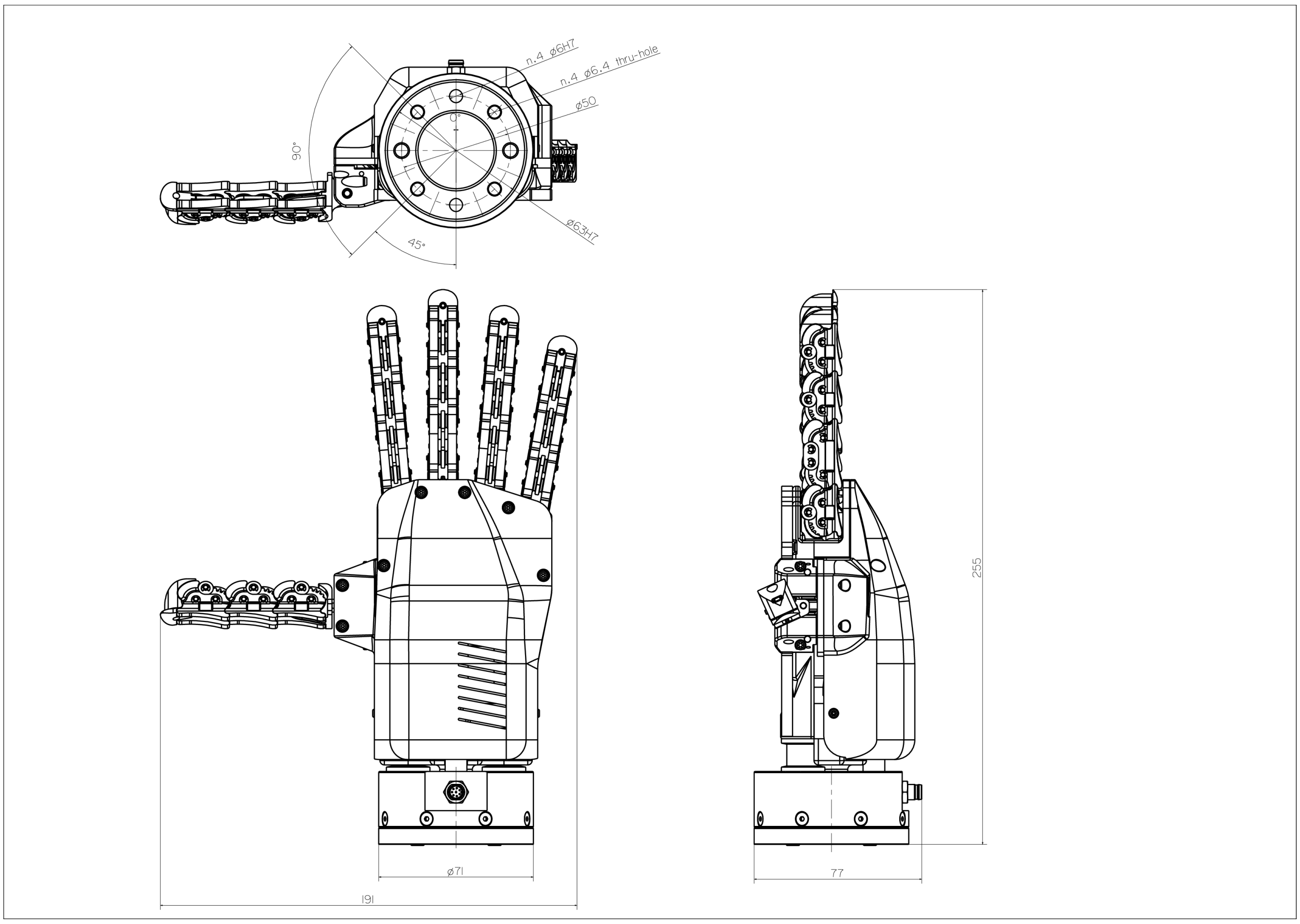


Figure 1. — qb SoftHand2 Research drawings

WARNING:

Be aware that the above figure may be out of scale.

3.2 Center of Mass

The following table gives also the position of the Center of Mass of the SoftHand2 Research and defines its Coordinate System, like depicted in the following schemes.

	UNIT	LEFT	RIGHT
Hand opened, CoM x coordinate, O_{TX}	[mm]	-3.8	3.8
Hand opened, CoM y coordinate, O_{TY}	[mm]	96.0	96.0
Hand opened, CoM z coordinate, O_{TZ}	[mm]	1.5	1.5
Hand closed, CoM x coordinate, O_{TX}	[mm]	-2.6	2.6
Hand closed, CoM y coordinate, O_{TY}	[mm]	85.5	85.5
Hand closed, CoM z coordinate, O_{TZ}	[mm]	2.5	2.5
Weight	[kg]	0.940	

Center of Mass, O_T, is evaluated w.r.t. Σ_M considering the hand mounted with its palm facing the ground, cf. the figures below.

NOTE:

All coordinate systems here described follow the orthogonal right-hand rule.

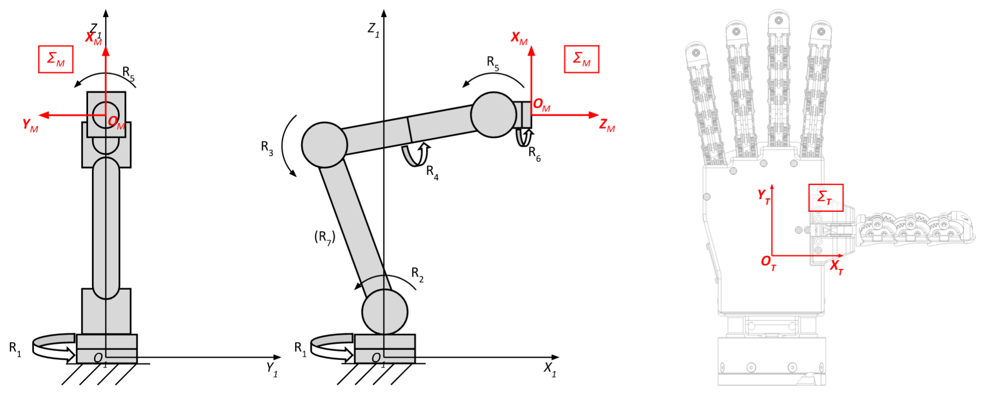


Figure 2. — Robot Tool Coordinate System

On the left, it is represented the mechanical interface coordinate system Σ_M *(O_M; X_M, Y_M, Z_M)* of an articulated robot, as defined by ISO 9787:2013. In particular, the center of the coordinate system, O_M , is on the interface surface of the robot tool flange and the Z_M axis is coincident with the tool flange axis of symmetry. R_i is the i-th revolute joint of the robot and *(O_1; X_1, Y_1, Z_1)* is the base coordinate system of the robot.

On the right, there is the Tool Coordinate System, Σ_T (O_T; X_T, Y_T, Z_T), of the SoftHand. The center, O_T , coincides with the center of mass of the hand and its position is defined by the distance from O_M . The Z_T axis is normal to the palm, outgoing positive, and the Y_T axis follows the proximal-distal direction and orientation. This definition implies that Σ_T is the same for left and right SoftHands. So, the positive orientation of X_T axis will be on the side of the thumb in the right hand and on the side opposite to the thumb in the left hand.

3.3 Mechanical characteristics

		BEST VALUE
Pinch grasping payload	[kg]	2.0
Power grasping payload	[kg]	3.0
Open pinch grasping payload	[kg]	30
Manipulation torque	[Nmm]	50
Index pushing force	[N]	0.5
Full closing time	[s]	1.0
Look at the graph for details.		

For this SVG the export flag is not set!

For this SVG the export flag is not set!

For this SVG the export flag is not set!

3.4 Electrical characteristics

		MINIMUM	NOMINAL	MAXIMUM
Power supply voltage	[VDC]	23.1	24	24.8
Power consumption	[W]	2.5	22	38
Current limit	[A]	—	0.5	*
Operational time	[s]	—	—	300
Duty cycle	[%]	—	—	69
Operating temperature	[°C]	-5	20	50
Relative humidity **	[%]	5	—	80
Typical noise level	[dB]	44	62	72

** Starting spike current may be greater, but it is instantaneous.*
*** Only the non-condensing case is to be considered.*

3.5 Tool connector pinout

On the qb SoftHand flange there is a Phoenix Contact 8-position M8 connector, A-coded, with gold-plated copper alloy contacts — Part Number: 1424232[1].

The following cables are compliant and recommended:

- Phoenix contact: `3m` free cable end with angled socket — Part Number: 1404192[2]
- Phoenix contact: `300mm` female/female with angled sockets — Part Number: 1098315[3]

The RS485 protocol characteristics are as follow:

- 8 bit;
- no parity;
- 1 stop bit;
- no flow control;
- 2M baud rate.

1 https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=1424232
2 https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=1404192
3 https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=1098315

#	WIRE COLOR	DESCRIPTION
1	White	RS485+
2	Brown	RS485-
3	Green	—
4	Yellow	—
5	Gray	24V ***
6	Pink	GND ***
7	Blue	24V ***
8	Red	GND ***

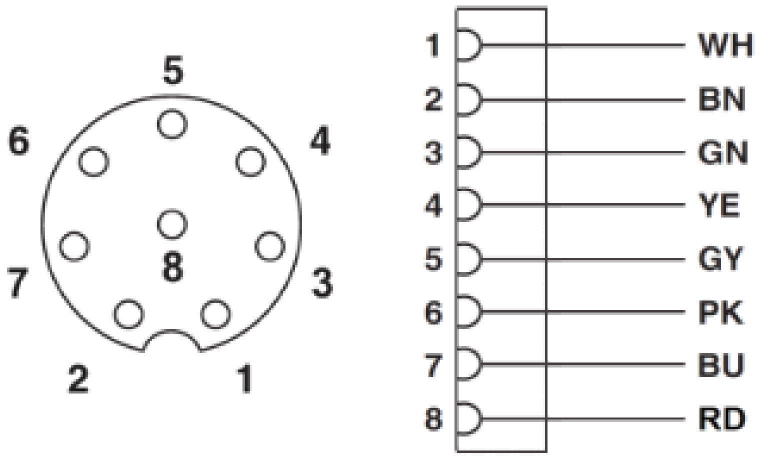


Figure 3. — User view

*** *24VDC and GND must be provided by an isolated Power Supply Unit.*

WARNING:

Any error in the pins connection may cause irremediable damages to the device.

4 Description

4.1 General

qb SoftHand2 Research is an anthropomorphic robotic hand based on soft-robotics technology. It is flexible, adaptable and able to interact with the surrounding environment, objects, or even humans while limiting the risk of injuring operators, spoiling products, or damaging the robot itself.

The introduction of the two-motors attuation allows the qb SoftHand2 Research to exploits the principles of first and second synergies in an intrinsically intelligent design, that is not only safe w.r.t. unexpected human-robot interaction but also adaptable to grasp and manipulate different shaped objects, showing an unparalleled level of simplicity and flexibility. Combining the two Synergies and/or the movements of motors, the qb SoftHand2 Research can performs different gestures, as shown in the following picture.

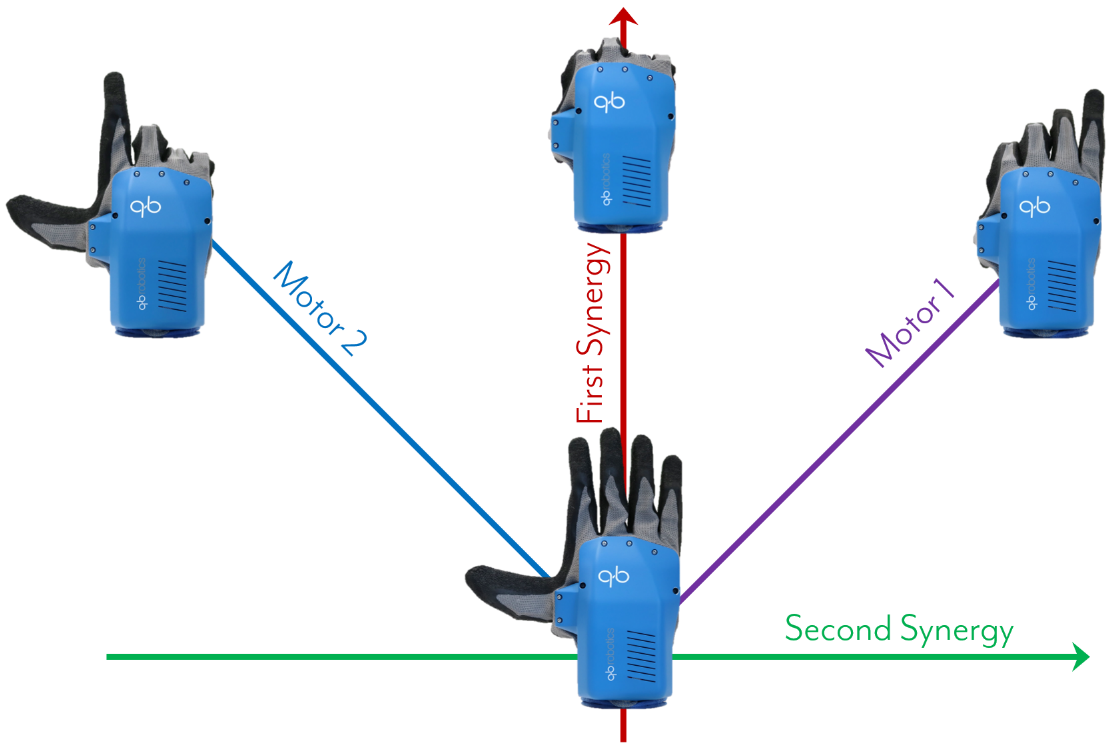


Figure 4. — qb SoftHand 2 Research gestures.

The combination of these innovations results in a plug-and-play, simple to control and affordable device.

The custom-made electronic board inside the qb SoftHand2 Research is composed of a logic stage for communication and low-level computation, and a power stage for motion control. This, together with a DC motor and its absolute encoder, establishes a simple position and current control feedback regulated by a properly tuned PID controller.

The qb SoftHand2 is generally shipped with an Adapter Board and a 3 meter-length cable.

The qb SoftHand2 Research does not use Common Industrial Protocols (CIP) or other industrial standards for I/O communication due to historical design and technical reasons.

I/O data basically refers as measurements from the device, commands to the device, and parameters from/to the device. These are handled in a custom package format as follows

```
2-byte — common preamble;
1-byte — target slave id;
1-byte — payload length;
n-byte — payload;
1-byte — checksum for communication integrity.
```

A custom-made single-master-multiple-slave (SMMS) serial communication protocol is implemented to:

- send commands to the connected devices;
- read measurements from the connected devices (motors position and/or currents);
- get and set configuration parameters.

4.2 Characteristics and key features

- Flexibility, adaptivity and robustness thanks to the soft-robotics design;
- 19 anthropomorphic DOFs controlled in two synergies motion;
- Dislocatable and self-reposition phalanges;
- Up to 50 Nmm manipulating torque*;
- Up to 2.0 kg maximum payload*;
- Maximum closure time of 1.0 s;
- Total weight of 940 g (including aluminium flange and screws);
- ROS packages and general C++ API available;

** Manipulating torque and payloads highly depend on object dimensions and approaching strategy (cf. Mechanical Characteristics* (see Page 6) *).*

4.3 Standard customization

- Mechanical flange adapters for non-standard robot flanges;
- Left and Right chirality configurations.

NOTE:

We may evaluate other customization for special partners; for requests, please contact our sales team[4].

4 mailto:sales@qbrobotics.com

5 Mounting and wiring

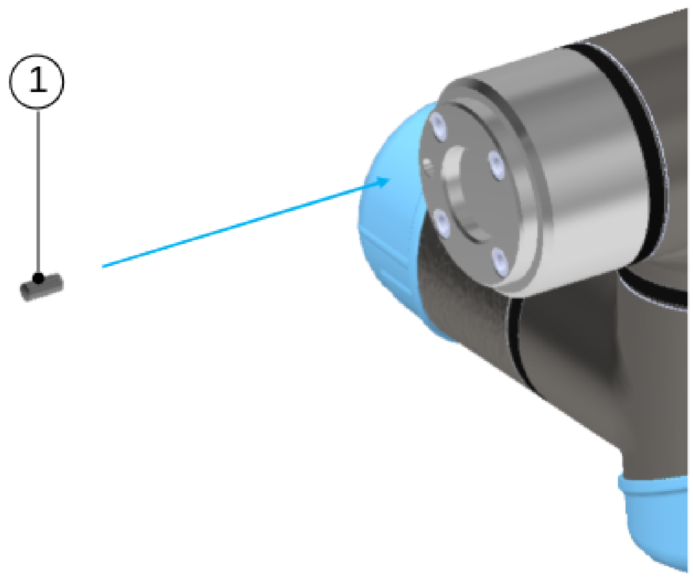
5.1 Tool mounting

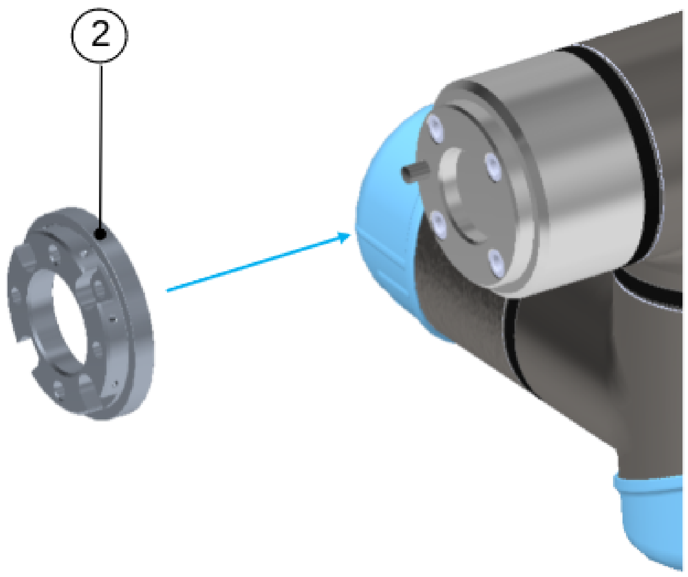
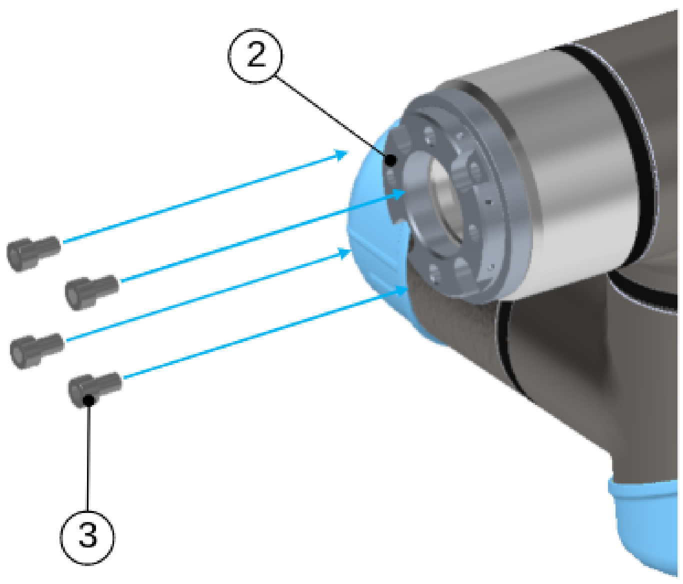
The qb SoftHand2 Research kit allows you to connect the device to the robot arm.

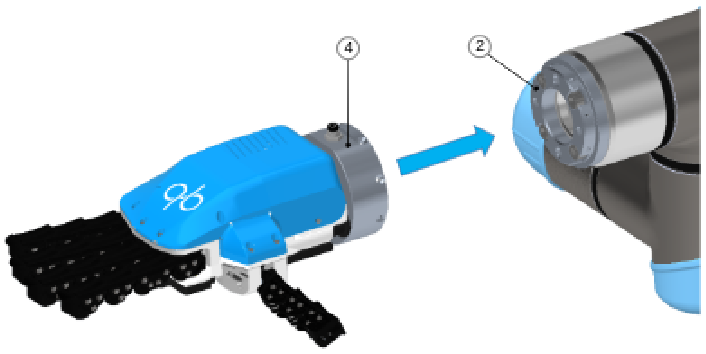
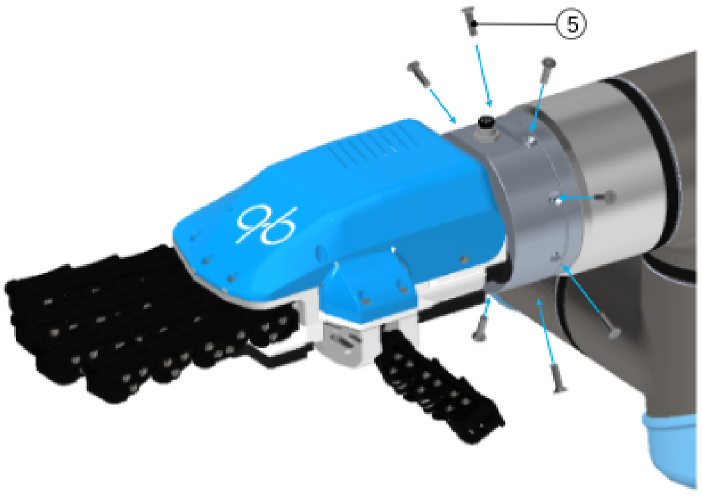
The device can be mounted on any robot equipped with a mounting interface ***ISO 9409*-1**.

⚠ Use only the screws provided within the package. Longer screws could damage the robot or the hand!

To assemble the hand on the robot arm, please follow the following instructions:

	INSTRUCTIONS
1	Insert (1) into the 6 mm hole on the wrist. 

	INSTRUCTIONS	
2	Center (2) on the wrist diameter 63 mm, taking care that the cylindrical pin (1) fits into one of the four d6 reamed through holes.	
3	Fasten (2) to the robot wrist by tightening the four screws (3). You need the 5 mm Allen wrench.	

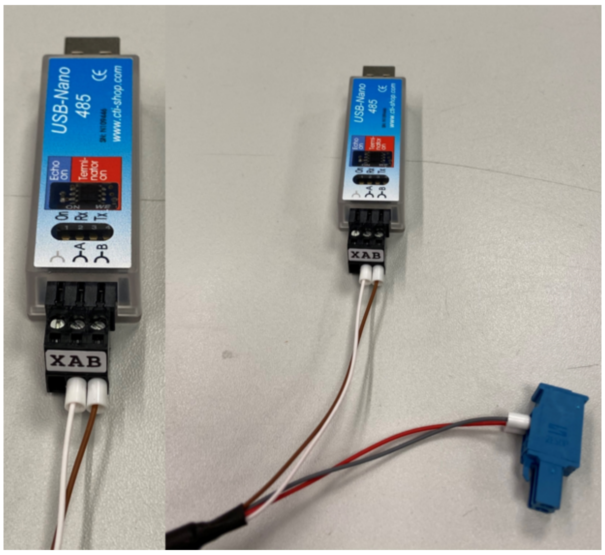
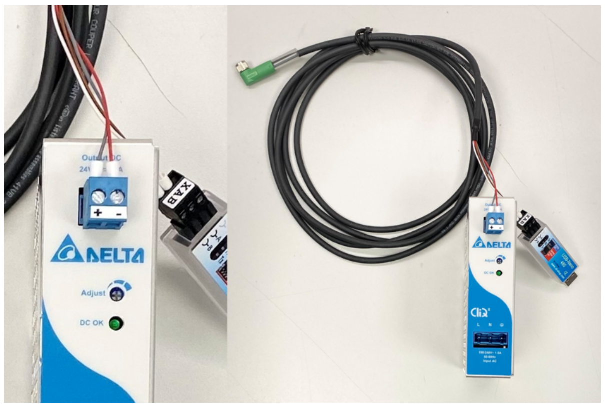
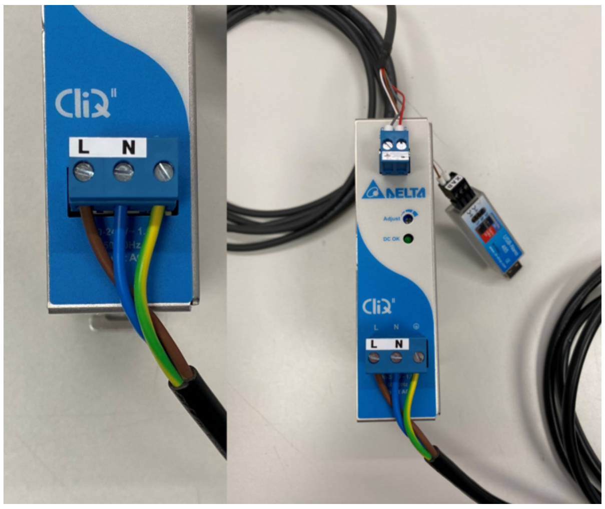
	INSTRUCTIONS	
4	Center (4) on the diameter 63 mm of (2), taking care that the plug fits into one of the four d6 mm through holes.	
5	Fix axially the hand by tightening the 8 screws (5). You need the 2 mm Allen wrench.	

5.2 Cable connections

To power and connect the qb SoftHand2 Research to UR robots' control box or a workstation computer, please follow the instructions in the table, using the provided kit.

> ⚠ The following images include an external power supply kit **(sold separately)**. If your robot's control box allows to power external devices, it is not necessary.
> In this case, please read the following steps carefully.

For this SVG the export flag is not set!

INSTRUCTIONS		
1	Connect the main cable (C1) to the RS485-USB converter (B).	
2	Connect the main cable (C1) to a power supply unit (A) (**Not connected to the power**). If you have not required the external power supply, the C1 cable has two unconnected red and grey ends. These are used to power the device, so connect the **red wire to the negative (V$^-$) and the grey wire to the positive connector (V$^+$).** For example in UR CB series robots you can connect the device to 24 V sockets inside the Digital Inputs panels.	
3	If you have purchased our external power supply, connect the power cable (C3) at the power supply unit (A).	

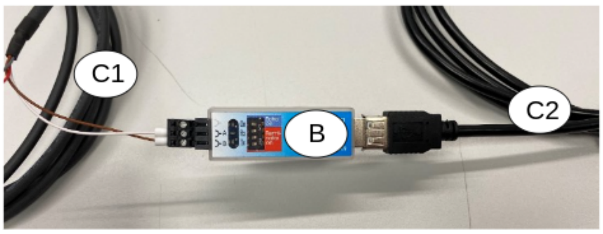
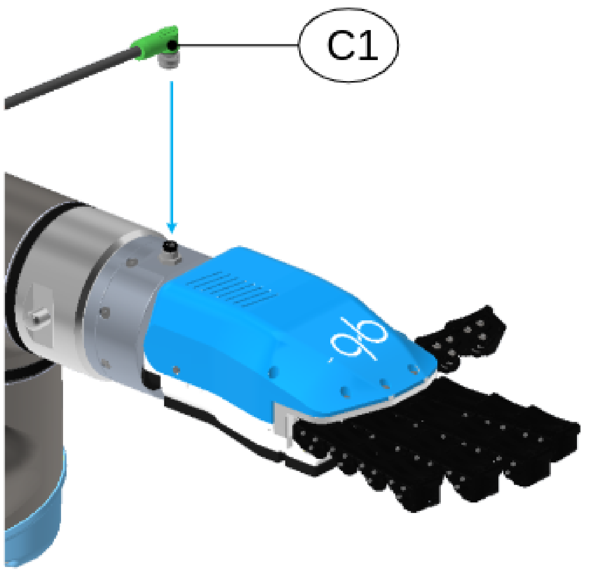
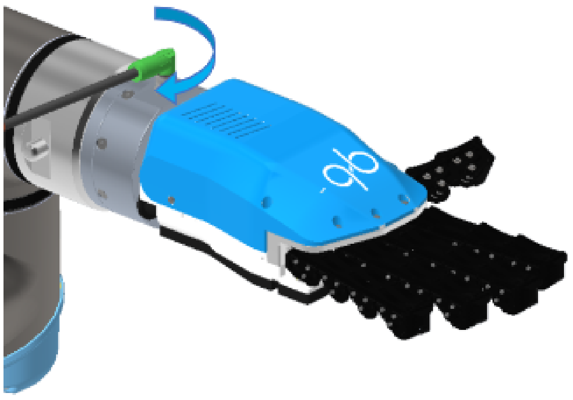
	INSTRUCTIONS	
4	Connect the USB cable (C2) at the RS485-USB converter (B); then connect the other end to a USB type-A port on your PC. ⚠ The qb SoftHand2 Research has to be used only via C++ APIs or ROS packages, therefore the USB communication cable has to be connected to the workstation PC: do not connect it to the robot control box.	
5	Insert the M8 connector of (C1) into the receptacle on the hand wrist flange. Lock the M8 connector by manually tightening its threaded ring on the body of the receptacle (**max tightening torque**= **0.2 Nm**). ⚠ Be careful to insert the connector in the correct direction. Do not rotate the M8 connector when inserted into the receptacle. Minimum curve radius of the cable (C1) = 51 mm.	 

Table 1. — Installation guidelines of the SoftHand2 Research.

⚠ YELLOW and GREEN wires on (C1) cable are optional. They are used to enable the button on the qb SoftClaw wrist flange as a Digital Input for the robot on which the gripper is mounted.
If your robot provides digital inputs, you can connect one of these wires (for example the yellow one) to a digital input port. The other wire (for example the green one) should be connected to the DC voltage output that your robot needs to detect the logic levels of the digital I/O.
Contact qbrobotics Support Team[5] for more informations.

5 mailto:support@qbrobotics.com

6 Software

6.1 Configuration GUI

6.1.1 Installation

You can download our GUI from our website[6] (download area).

To install all the requirements for your specific operating system, please refer to the how-to section (see Page 39) .

⚠ For Linux users: use `chmod +x` on a the downloaded file and make it executable. Right click on your script and chose **Properties** -> **Permissions** -> **Allow executing file as program**, leaves you with the exact same result as the command in terminal.

6.1.2 Usage

The *qb SoftHand2 Research* GUI (generally referred as *qbtools*) is a simple application to quickly test the basic functionalities of the device through a personal computer. It may be also useful to diagnose — together to our support team[7] — troubles about hardware or software components.

The very first thing to do is to click `"Scan Ports"` at the top of the window. If a qb device is properly connected to the system, its serial port name is shown in the drop-down menu on the left.

⚠ If there are more than one serial port available, be sure to select the one connected to the qb SoftHand2 Research of interest.
If no serial port is found, check out the installation (see Page 39) and mounting (see Page 13) steps.

When a serial port is found, the "Connect" button becomes available and by clicking on it you scan the serial resource to find every qbrobotics device connected to the system. If there is at least one device found, this operation enables all of the GUI features and it shows a green Connected label next to the button.

The button `Activate` activates or deactivates the motor driver on the *qb SoftHand2 Reserach*; the current activation status is shown next to the Active/Inactive button;

⚠ Remember to power the *qb SoftHand2 Research* before using this GUI.

6 https://qbrobotics.com/product/qb-softclaw/
7 mailto:support@qbrobotics.com

⚠ Two devices connected to the same master PC must never have the same ID value. Unexpected behaviors may happen otherwise.

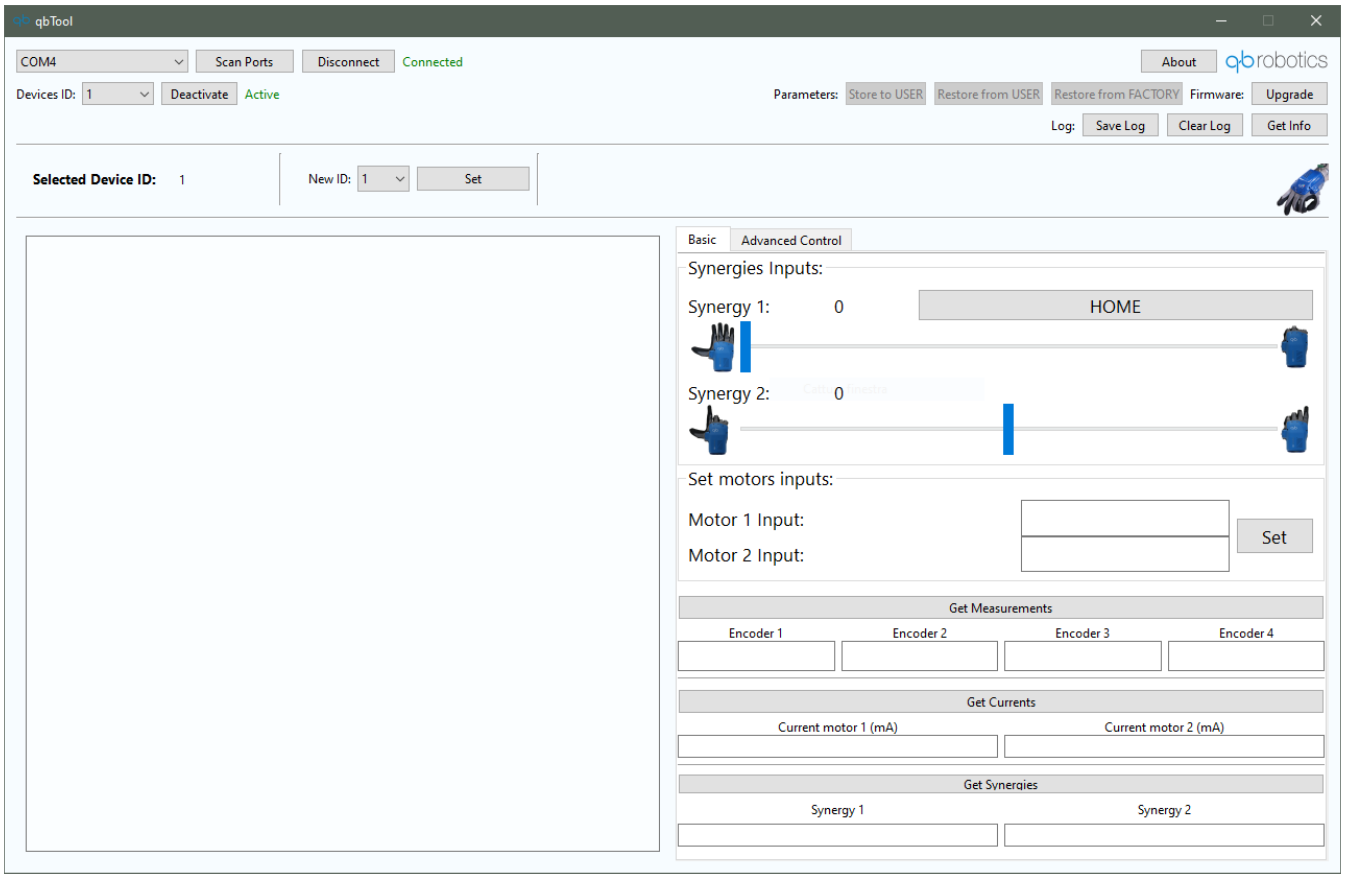
The button of the tab allow:

- `About` : Info about the current GUI.
- `Upgrade` : upgrade the qbSoftHand2 Research firmware. Contact the support team[8] for information about the device upgrade.
- `Save Log` : save the device log in a file.
- `Clear Log` : clear the information printed in the panel.
- `Get Info` : shows useful information about the qb SoftHand in the panel. It prints the current state and its settings, e.g. firmware version, parameter values, measurements...
- `Devices ID:` shows a list of device IDs connected to the system; by selecting one of them from the drop-down menu you are able to query that specific device — i.e. requesting information from and sending commands to it;
- `Selected Device ID:` shows the device ID that is currently connected and usable;
- `New ID:` sets a new ID value from the related drop-down menu to the connected qb device; `"Current ID"` is updated accordingly;

Tabs are described in the following section.

8 mailto:support@qbrobotics.com

Basic tab


COM4 Connected

Devices ID: 1 Active

Selected Device ID: 1

New ID: 1

Synergy 1: 0

Synergy 2: 0

Figure 5. — qb SoftHand2 Research GUI screen - basic tab

The motors must be activated first to be able to send references to the them.

- `Measurements and Currents` : the three buttons `“Get Measurements”` , `“Get Currents”` and `“Get Synergies”` enable an auto-refreshing threads that shows respectively the encoder measurements, the motor current (in milliamperes) and the actual synergies;
- `Sliders` : It is possible to set the synergies to device.
- `Inputs` : Set directly reference signal to the device’s motors.
- `Home` : moves the device to start position.

Advanced Control tab

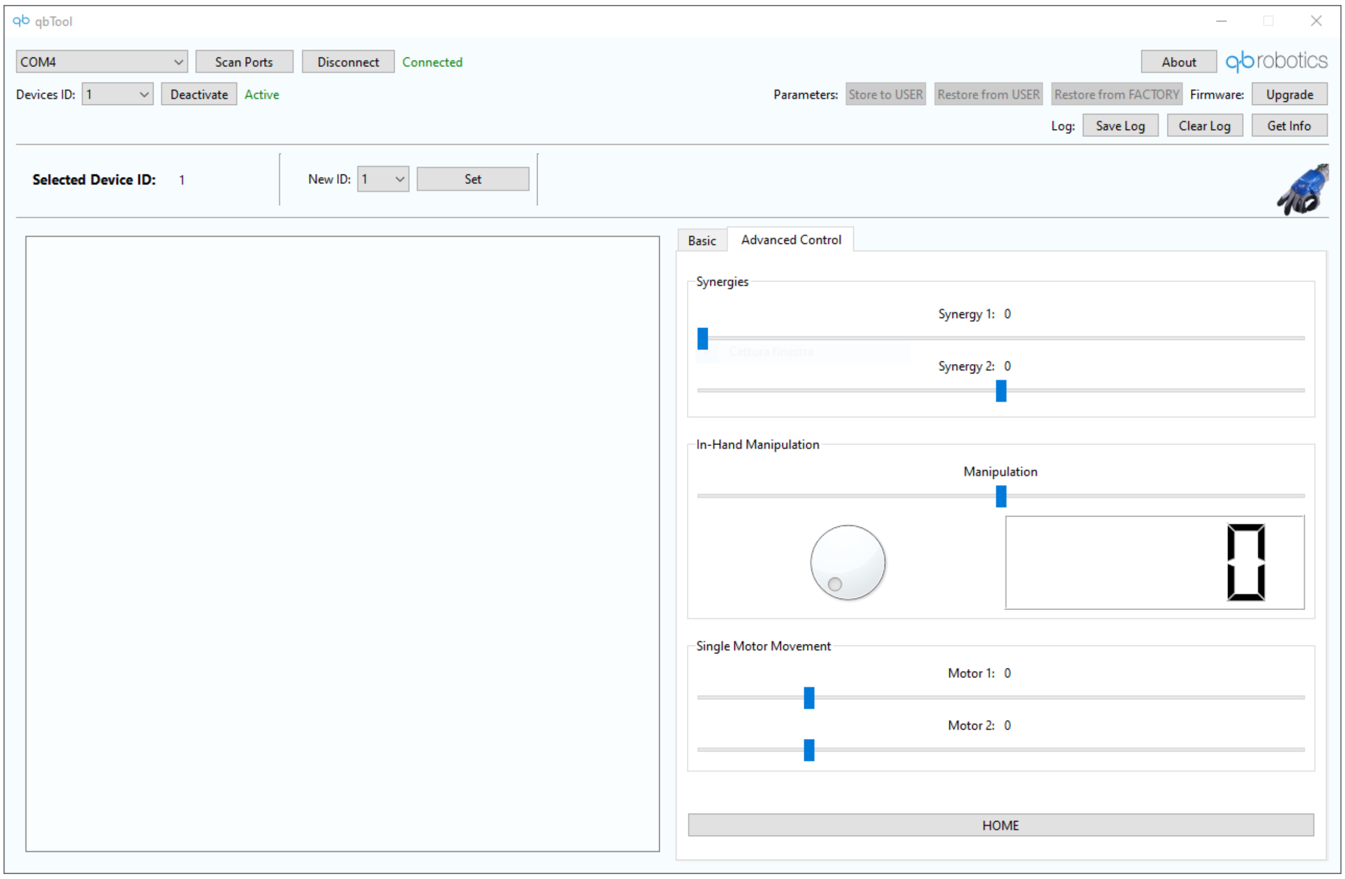


Figure 6. — qb SoftHand2 Research GUI screen - advanced control tab

This tab is intended to give the user a better understanding of how the device works.

- `Synergies sliders` : It is possible to set the synergies to the device.
- `Manipulation slider and dial` : It is a simple example of how to control motors in order to perform object manipulation. Using this slider, one motor moves in the opposite direction of the other but not of the same quantity.

This is not a control modality but just an example of how to control the motors. Implement your own manipulation strategy!

- `Motor sliders` : It is possible to set the motor references to the device.
- `Home` : moves the device to start position.

6.2 ROS

6.2.1 Installation

If you have never set it up, you probably need to add your linux user to the `dialout` group to grant right access to the serial port resources. To do so, just open a terminal and execute the following command:

```
sudo gpasswd -a <user_name> dialout
```

where you need to replace the `<user_name>` with your current linux username.

NOTE:

Note: don't forget to logout or reboot.

Sources

ROS packages are available and open source. It is possible to download them from the bitbucket repository of qbrobotics[9].

The *qbSoftHand2 Research* requires the following ROS packages versions (or newer):

- qbdevice-ros: 3.0.4
- qbhand-ros: 3.0.1

WARNING:

Since you are interested in the ROS interfaces for our devices, it is assumed that you are familiar at least with the very basics of the ROS environment. If not, it might be useful to spend some of your time with ROS[10] and catkin[11] tutorials. After that, don't forget to come back here and start having fun with our Nodes.

Install the *qb SoftHand2 Research* packages for a ROS user is straightforward. Nonetheless it is necessary to pay attention to some steps when repositories are cloned.

The ROS package that is used to control the qbrobotics devices (qbdevice-ros[12]) contains within the C++ API as **submodules**. The following are the detailed steps which should be easy to understand even for ROS beginners:

1. Clone both the `qb_device` and `qb_hand` packages to your Catkin Workspace, e.g. `~/catkin_ws` :

9 https://bitbucket.org/qbrobotics/
10 https://wiki.ros.org/ROS/Tutorials
11 https://wiki.ros.org/catkin/Tutorials
12 https://bitbucket.org/qbrobotics/qbdevice-ros/src/production-melodic/

```
cd ~/catkin_ws/src
git clone --recurse-submodules https://bitbucket.org/qbrobotics/qbdevice-ros.git
git clone https://bitbucket.org/qbrobotics/qbhand-ros.git
```

2. Compile the packages using `catkin`:

```
cd ~/catkin_ws
catkin_make
```

If you were not familiar with ROS you should be happy now: everything is done! Nonetheless, if you encounter some troubles during the compilation, feel free to ask for support on our Bitbucket[13].

NOTE:

Depending on your ROS installation, you may need some extra packages to properly compile the code. Please, be sure that you have already installed at least `ros-<ros_distro>-ros-controllers`, `ros-<ros_distro>-transmission-interface`, `ros-<ros_distro>-joint-limits-interface`, `ros-<ros_distro>-combined-robot-hw`, `ros-<ros_distro>-rqt-joint-trajectory-controller` and their dependencies (*e.g.* *use* `sudo apt install <ros-pkg>`).

Device Setup

Connect a *qb SoftHand2 Research* to your system is basically a matter of plugging in a USB cable. Nonetheless, **read carefully** the manual to understand all the requirements and advice about either single-device or chained configurations.

6.2.2 Usage

As shown in the following picture there are two distinct configurations to control several *qb SoftHand2 Research* devices connected to the system:

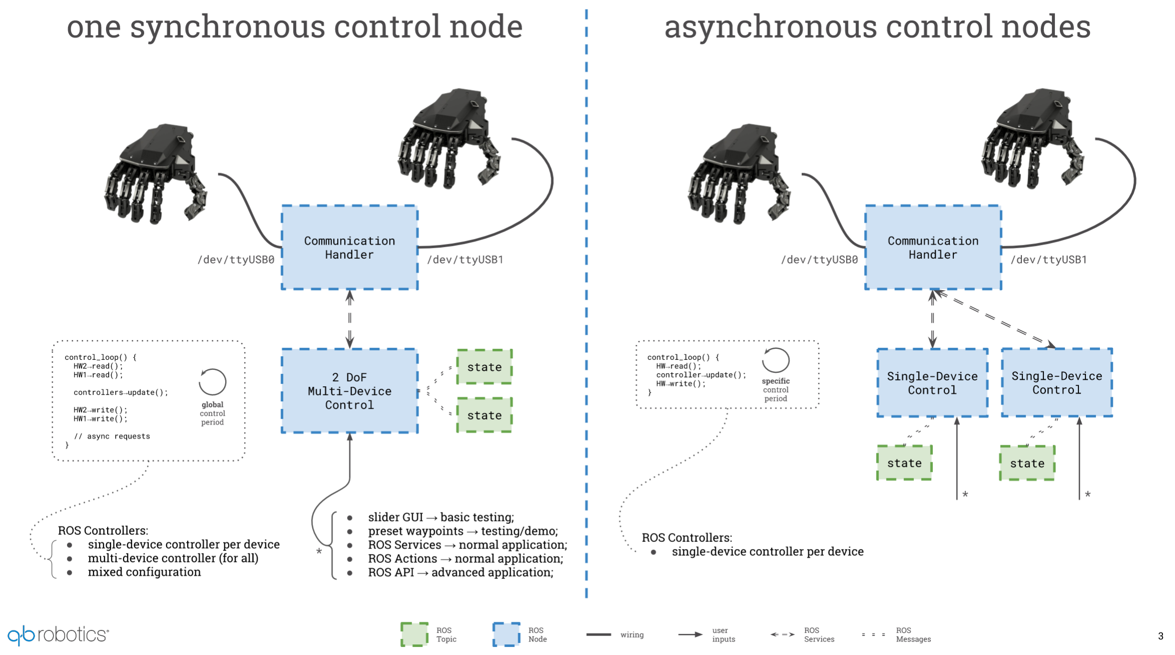
- The first (and recommended) groups all the Hardware Interfaces together (thanks to the combined_robot_hw[14]) and exploits them as a unique robot system. We have called it *"synchronous"* just to point out that every sequence of reads and writes is always done in the same predefined order.
- The second mode threats every device as an independent Hardware Interface with its dedicated ROS Node which executes the control loop independently w.r.t. the rest of the system, i.e. *"asynchronously"*.

Mixed configurations can be also achieved through a proper setup. In such a case we can think of synchronous sub-systems which execute asynchronously w.r.t. each other.

13 https://bitbucket.org/qbrobotics/qbhand-ros/issues?status=new&status=open
14 https://wiki.ros.org/combined_robot_hw

NOTE:

In a single-device system the synchronous mode is a nonsense.


one synchronous control node
asynchronous control nodes
Communication Handler
/dev/ttyUSB0
/dev/ttyUSB1
2 DoF Multi-Device Control
state
state
global control period
Single-Device Control
Single-Device Control
specific control period
*
ROS Controllers:
single-device controller per device
multi-device controller (for all)
mixed configuration
slider GUI → basic testing;
preset waypoints → testing/demo;
ROS Services → normal application;
ROS Actions → normal application;
ROS API → advanced application;
ROS Controllers:
single-device controller per device
qbrobotics
ROS Topic
ROS Node
wiring
user inputs
ROS Services
ROS Messages
3

```
control_loop() {
  HW2→read();
  HW1→read();

  controllers→update();

  HW2→write();
  HW1→write();

  // async requests
}
```

```
control_loop() {
  HW→read();
  controller→update();
  HW→write();
}
```

Figure 7. — qb SoftHand ROS control mode schemes

In both cases there is always one central Node which manages the shared resources for the serial communication (e.g. one or many USB ports) and which provides several ROS services to whom wants to interact with the connected devices. This Node is called *Communication Handler*.

CRITICAL:

Please remember that in a multi-device configuration, each *qbrobotics* device connected to your system **must have a unique ID**. In order to properly set it, you have to use our GUI (see Page 19) .

Details

To understand what is hiding under the hood, have a look at the C++ classes overview which sums up all the main concepts of our ROS packages:

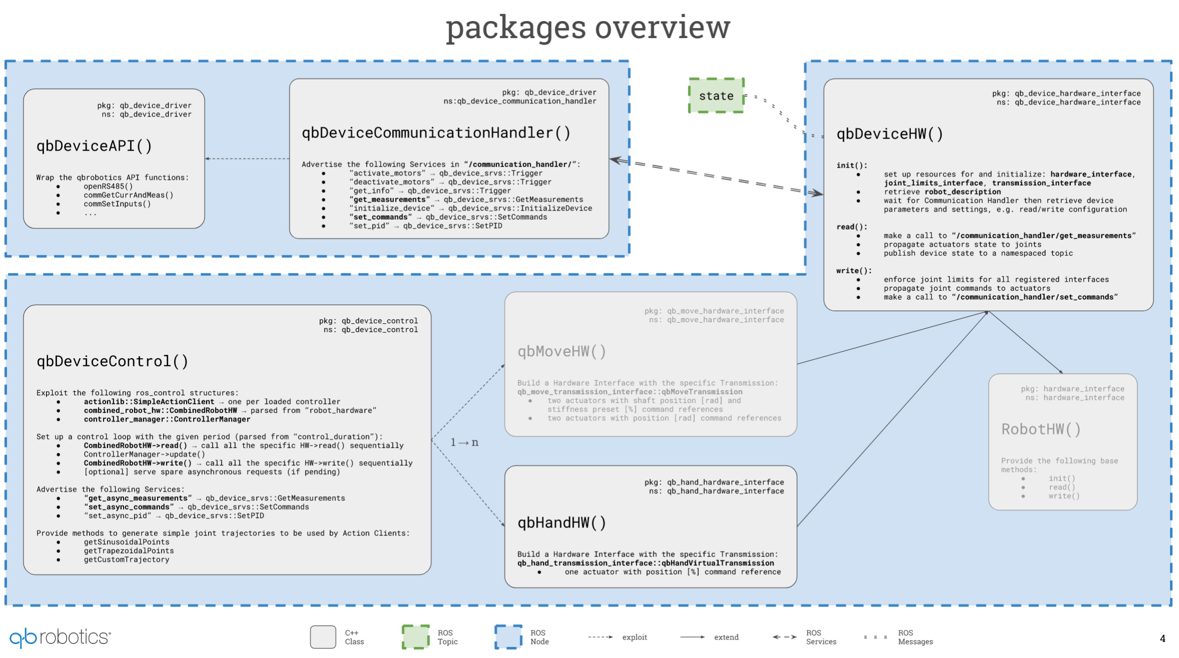


Figure 8. — qb SoftHand ROS packages and classes overview

Communication Handler

The Communication Handler Node has no parameters to be set, therefore it is always launched like this:

```
roslaunch qb_device_driver communication_handler.launch
```

On start, it scans the serial communication resources connected to your system and shows a list of the devices it has found. By default, it never scans again for new devices, apart from asking it explicitly during the initialization of a control Node.

This is a simple example when starting the Communication Handler with two *qbrobotics* devices connected on two distinct USB ports:

```
[ INFO] [1657115843.575789176]: [CommunicationHandler] Found 1 qbrobotics devices on [/dev/ttyUSB0]
[ INFO] [1657115843.776596083]: [CommunicationHandler] The device with id 1 is connected.
[ INFO] [1657115844.108091910]: [CommunicationHandler] Found 1 qbrobotics devices on [/dev/ttyUSB1]
[ INFO] [1657115844.308228280]: [CommunicationHandler] The device with id 2 is connected.
```

When the Communication Handler is on, it provides all the Services required to interact with the connected devices: e.g. *get info or measurements, activate or deactivate motors, set commands*, and even more... A detailed description of the services can be found in the qb_device_driver[15] package wiki.

15 https://wiki.ros.org/qb_device_driver

Control

As shown before, the control Node exploits the ros_control[16] Controller Manager which loads and runs the device controllers. Each controller provides an Action Server that, together with the Hardware Interface structure, allows the user to send commands to the relative device and get its measurements.

From an API point of view, it is implemented an Action Client which matches the relative trajectory controller and provides a method to send Goals, i.e. command references, directly to the given device. Additionally the Action Client is subscribed to a Topic (`*_controller/command`) that can be used to send reference commands from outside the code, e.g. asynchronously from the command line, or from a higher level control Node, e.g. as a result of a planning algorithm.

WARNING:

It is recommended not to mix these two control modes: choose either to control the device directly from the code by extending our API or through this command Topic.

Regardless the control mode chosen for the given application, and apart form a customization of the API, an example of how to launch and control one or more *qb SoftHand2 Research* with one synchronous control node can be found in the subpackage qb_hand_control, in the launch folder. The files **control_qbhand2m.launch** or **control_qbhand2m_chain.launch** explains how to control one or several devices.

Control Modes

For the sake of simplicity, we are going to cover all the control modes for a single *qb SoftHand2 Research*, but it is just a matter of putting things together and set the launch file parameters properly to control several devices together.

All the control modes are initialized in the same manner but with distinct command line arguments. The default single-device control Node which brings everything up and simply waits for commands on the above-mentioned Action topic is started with the following command:

```
roslaunch qb_hand_control control_qbhand2m.launch activate_on_initialization:=true standalone:=true
use_controller_gui:=true
```

The arguments explained

- `activate_on_initialization [false]`: Activates the motor at startup (the device will not move since the first command reference is received).
- `device_id [1]`: Each device has its own ID, you need to set the one of the actual device connect to your system.
- `standalone [false]`: Starts the Communication Handler together with the control Node. If you set this to `false` (or remove it since the default value is `false`), you need to launch the Communication Handler in a separate terminal.

16 https://wiki.ros.org/ros_control

It is worth noting that the activation of the motor can be postponed to improved safety if you are not aware of the state of the system at startup. To do so just set `activate_on_initialization:=false` (or remove it since the default value is `false`) and make a call to the Communication Handler `activate_motors` Service, when your system is ready, e.g. as follows:

```
rosservice call /communication_handler/activate_motors {"id: <actual_device_id>, max_repeats: 2"}
```

Additional arguments

- `control_duration [0.01]` : The duration of the control loop expressed in seconds.
- `get_currents [true]` : Choose whether or not to retrieve current measurements from the device.
- `get_positions [true]` : Choose whether or not to retrieve position measurements from the device.
- `get_distinct_packages [true]` : Choose whether or not to retrieve current and position measurements from the device in two distinct packages.
- `max_repeats [3]` : The maximum number of consecutive repetitions to mark retrieved data as corrupted.
- `set_commands [true]` : Choose whether or not to send command positions to the device.
- `set_commands_async [true]` : Choose whether or not to send commands without waiting for ack.
- `use_rviz [false]` : Choose whether or not to use rviz. If enabled you should see a virtual hand on screen performing a similar behavior.

⚠ The rviz visualization is able to show only the device behavior when the first synergy is controlled.

- `use_without_robot [true]` : Choose whether or not to bring up robot description (e.g. when using with other robots is better to set it apart).

ⓘ Be aware that the *qb SoftHand2 Research* is desensorized and therefore it is not possible to know exactly the position of each finger: the screen visualization is just the result of an estimation of the closure value and may differ from the real configuration of your *qb SoftHand* (e.g. when grasping an object).

Two different controller has been implemented within the qbrobotics ROS package for the qbSoftHand2 Research:

- The `synergies_trajectory_controller` : using this controller it is possible to control directly the two synergies (named `synergy_joint` and `manipulation_joint`);
- The `motor_positions_trajectory_controller` : using this controller it is possible to control directly each motor, named `motor_1_joint` and `motor_2_joint` . The first motor controls the part

of the hand that includes the *thumb*, the second one controls the part of the hand that includes the *little finger*.

It is possible to switch between controllers with the following terminal command:

```
rosservice call /qbhand2m1/control/controller_manager/switch_controller "start_controllers: ['qbhand2m1_motor_positi
ons_trajectory_controller']
stop_controllers: ['qbhand2m1_synergies_trajectory_controller']
strictness: 0
start_asap: false
timeout: 0.0"
```

The previous command can be executed only after launching the qb SoftHand2 Research node.

The previous command works with a device with **ID 1**. If a device with different id is used you have to change the topic namespace. You can find the topic you need by running the command `rosservice list` after the node is started.

The following are two examples of terminal commands useful to test the device behavior when:

- The `synergies_trajectory_controller` is used (the qbSoftHand 2 Research closes completely):

```
rostopic pub -1 /qbhand2m1/control/qbhand2m1_synergies_trajectory_controller/command trajectory_msgs/
JointTrajectory "header:
 seq: 0
 stamp:
   secs: 0
   nsecs: 0
 frame_id: ''
joint_names: [qbhand2m1_manipulation_joint, qbhand2m1_synergy_joint]
points:
 - positions: [0, 1]
   velocities: [0, 0]
   accelerations: [0, 0]
   effort: [0, 0]
   time_from_start: {secs: 1, nsecs: 0}"
```

- The `motor_positions_trajectory_controller` is used (the qbSoftHand 2 Research perform an index gesture):

```
rostopic pub -1 /qbhand2m1/control/qbhand2m1_motor_positions_trajectory_controller/command
 trajectory_msgs/JointTrajectory "header:
 seq: 0
 stamp:
     secs: 0
     nsecs: 0
   frame_id: ''
 joint_names: [qbhand2m1_motor_1_joint, qbhand2m1_motor_2_joint]
 points:
   - positions: [0, 0.3]
     velocities: [0, 0]
     accelerations: [0, 0]
     effort: [0, 0]
     time_from_start: {secs: 1, nsecs: 0}"
```

You can use the previous messages structure to send control messages to qb SoftHand2 Research. The following parameters are important to properly control the device:

- `positions` - accepts values in range [0,1]
- `time_from_start` - indicates the time for concluding the actions. It sets indirectly the velocity of the qb SoftHand2 Research motion.

The other parameters can remain as they are.

> The previous command works with a device with **ID 1**. If a device with different id is used you have to change the topic namespace. You can find the topic you need by running the command `rostopic list` after the node is started.

The followings are particular control modes which are enabled with few parameters, but the concepts of this paragraph hold for all of them.

1. GUI Control

This control mode is the one suggested to test that everything is working as expected. You are able to open and close the *qb SoftHand3 Research* interactively, but nothing more than this.

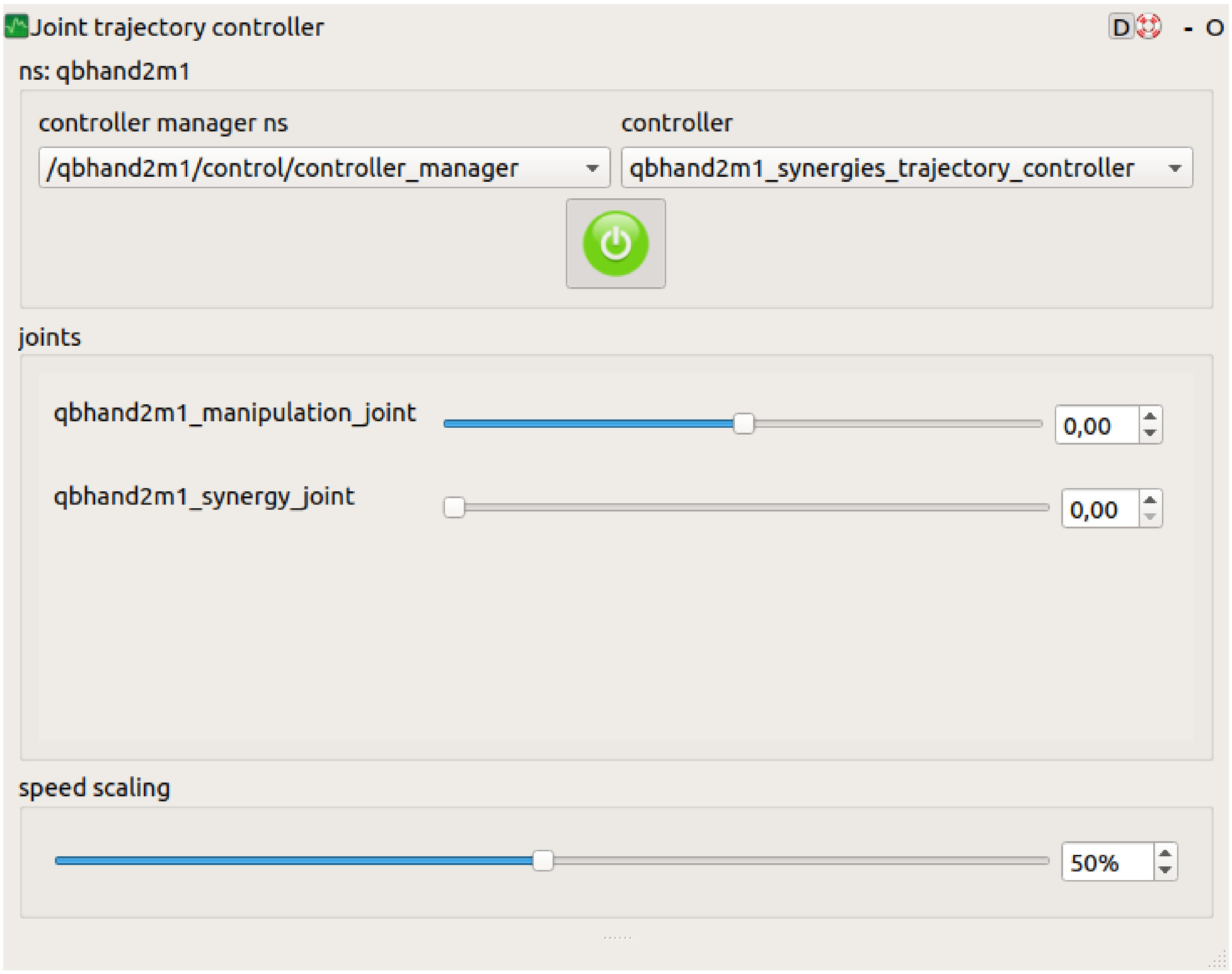
> You will probably need this only the very first times and for debugging.

To start this mode just add `use_controller_gui:=true` to the general `roslaunch` command (be sure that the opposite `use_waypoints` is not used).

After a while a GUI should appear to screen with two empty drop-down menus, a red enable button below them, and a *speed scaling* slider at the bottom.

1. Select the *Controller Manager* namespace from the left menu, e.g. `/<robot_namespace>/control/controller_manager` (where `<robot_namespace>` is an additional argument of the launch file needed with several devices). This enables the right menu which provides all the controllers available for the connected device.
2. Select the *qb SoftHand2 Research* running controller from the second drop-down menu and enable it through the circular button.
3. Two sliders will appear in the GUI to control the synergies (or the motors, depending on the running controller) of the hand. Move the sliders to control the *qb SoftHand2 Research*. You can also vary the

speed through the bottom *speed scaling* slider if you like a faster/slower motion. No other timing constraints can be set in this mode.


Joint trajectory controller

ns: qbhand2m1

controller manager ns: /qbhand2m1/control/controller_manager

controller: qbhand2m1_synergies_trajectory_controller

joints

qbhand2m1_manipulation_joint 0,00

qbhand2m1_synergy_joint 0,00

speed scaling 50%

Figure 9. — qb SoftHand ROS GUI control

2. Waypoint Control

This control mode is a bit more structured and useful than the previous: it allows to set a fixed trajectory of any number of position waypoints (with timing constraints) and set the robot to cycle infinitely on it (because of the loop it is recommended to set the first and last waypoint in a similar configuration to avoid unwanted sudden changes).

To start this mode just add `use_waypoints:=true` to the general `roslaunch` command (be sure that the opposite `use_controller_gui` is not used). You won't see any control interface in this case but the *qb SoftHand2 Research* should start moving according to the given trajectory, parsed from a yaml file located at

```
<robot_package>_control/config/<robot_name>_waypoints.yaml
```

where `robot_name` and `robot_package` are two additional launch file arguments.

Customization

You can modify the waypoint trajectory to replicate the behavior you want: either change the `<robot_package>_control/config/<robot_name>_waypoints.yaml` or add another custom application-specific file in the `config` directory. In the second case you need to set the argument `robot_name` properly when launching the command from the terminal.

The waypoint configuration is as follows:

```
waypoints:
  -
    time: [2.0]
    joint_positions:
      qbhand2m1: [0.0, 0.0]
  -
    time: [4.0]
    joint_positions:
      qbhand2m1: [0.48, 1.0]
  -
    time: [8.0]
    joint_positions:
      qbhand2m1: [0.48, 1.0]
  -
    time: [9.0]
    joint_positions:
      qbhand2m1: [0.48, 0]
  -
    time: [10.0]
    joint_positions:
      qbhand2m1: [0.0, 0]
```

The previous configuration works with a device with **ID 1**. If a device with different id is used you have to change the device name (i.e. `qbhand2m1` in the previous file).

3. API Control

If you need a complex (i.e. real) control application, e.g. the *qb SoftHand2 Research* is mounted on a robot which uses computer vision aid to grasp objects, the previous two control modes don't really help much. What we provide for real applications is the full ROS libraries to manage and control the *qb SoftHand2 Research*.

You have to dig into the qb_hand[17] package documentation and find what better suits for your needs, e.g. extend the `qbDeviceControl` class provided, or even redesign some of its parts by following an approach similar to ours.

NOTE:

Our recommendation is to use as much as possible our resources, classes and macros to help you while developing your application. Don't reinvent the wheel!

At last, if you come up with a something useful for the whole community, it will be amazing if you propose your improvement with a Pull Request in the package of interest on our Bitbucket[18].

6.2.3 Support, Bugs and Contribution

Since we are not only focused on this project it might happen that you encounter some trouble once in a while. Maybe we have just forget to think about your specific use case or we have not seen a terrible bug inside our code. In such a case, we are really sorry for the inconvenience and we will provide any support you need.

To help you in the best way we can, we are asking you to do the most suitable of the following steps:

1. It is the first time you are holding a *qb SoftHand2 Research*, or the first time you are using ROS, or even both: it is always a pleasure for us to solve your problems, but please consider first to read again the instructions above and the ROS tutorials. If you have ROS related questions the right place to ask is ROS Answers[19].
2. You are a beginner user stuck on something you completely don't know how to solve or you are experiencing unexpected behaviour: feel free to contact us at support@qbrobotics.com[20], you will receive the specific support you need as fast as we can handle it.
3. You are quite an expert user, everything has always worked fine, but now you have founded something strange and you don't know how to fix it: we will be glad if you open an Issue in the package of interest on our Bitbucket[21].
4. You are definitely an expert user, you have found a bug in our code and you have also correct it: it will be amazing if you open a Pull Request in the package of interest on our Bitbucket[22]; we will merge it as soon as possible.
5. You are comfortable with *qbrobotics* products but you are wondering whether is possible to add some additional software features: feel free to open respectively an Issue or a Pull Request in the package of interest on our Bitbucket[23], according to whether it is just an idea or you have already provided your solution.

In any case, thank you for using *qbrobotics*[24] solutions.

17 https://wiki.ros.org/qb_hand
18 https://bitbucket.org/account/user/qbrobotics/projects/ROS
19 https://answers.ros.org/questions/
20 mailto:support@qbrobotics.com
21 https://bitbucket.org/account/user/qbrobotics/projects/ROS
22 https://bitbucket.org/account/user/qbrobotics/projects/ROS
23 https://bitbucket.org/account/user/qbrobotics/projects/ROS
24 https://www.qbrobotics.com

6.3 C++ API library

6.3.1 qbdevice C/C++ API v6

6.3.2 Installation

To install all the requirements for your specific operating system, please refer to the how-to section (see Page 39) .

To compile the C/C++ API library:

- open a terminal (for Linux and macOS users), or the command prompt (for Windows users);
- navigate to the `"src"` directory inside the downloaded package;
- execute the `"make"` command.

Under `"lib_unix"` or `"lib_win"` directory you will find the newly created static library `"libqbmove_comm.a"` that can be linked in your own application.

NOTE:

Do not forget to include the `"qbmove_communications.h"` when linking the API library in your project.

Basic functions

In most cases these are all the necessary functions to integrate the *qb SoftHand2 Research* within your system.

- `int RS485ListPorts(char **serial_ports)` : retrieves all the serial ports connected to the system, returns their number, and fills the given argument with the connected serial port names;
- `void openRS485(comm_settings *file_descriptor, const char *serial_port)` : opens the serial communication by acquiring the serial port resource and sets the given file descriptor accordingly;
- `void closeRS485(comm_settings *file_descriptor)` : closes the serial communication and frees the serial resource;
- `void commActivate(comm_settings *file_descriptor, int device_id, char activate)` : activates (or deactivates) the motor on the *qb SoftHand2* with the given id, respectively if `"activate"` is `"0x03"` (or `"0x00"`);
- `int commGetActivate(comm_settings *file_descriptor, int device_id, char *activate)` : fills `"activate"` with the activation status of the motor on the *qb SoftHand2* with the given id, and returns 0 on success;

- `int commGetMeasurements(comm_settings *file_descriptor, int device_id, short int measurements[4])` : fills `"measurements"` with the position of the motor on the *qb SoftHand2* with the given id, and returns 0 on success.
- `int commGetCurrents(comm_settings *file_descriptor, int device_id, short int currents[2])` : fills `"currents"` with the current of the motor on the *qb SoftHand2* with the given id, and returns 0 on success.
- `void commSetInputs(comm_settings *file_descriptor, int device_id, short int inputs[2])` : sends a reference position command to the *qb SoftHand2* with the given id; the value is expressed in motor ticks which lay in range `"0, 23000"` .

CRITICAL:

These commands are meant to be used by expert users only.

6.3.3 qbdevice C++ API v7

qbdevice C++ API v7 are available on qbrobotics Bitbucket public repository[25]. The CMake[26] based project contains two *submodules*:

- **serial**: used to handle the communication with qbrobotics devices.
- **qbrobotics-driver**: used to control and set/get parameters from qbrobotics devices.

Cloning and compiling

In order to clone the project, it is necessary to run the following command:

```
git clone --recurse-submodules https://bitbucket.org/qbrobotics/qbdevice-api-7.x.x.git
```

Once the project is cloned, it is possible to compile qbrobotics libraries by executing the following commands in *qbdevice-api-7.x.x* folder:

```
mkdir build
cd build
cmake ..
make
```

When the compilation is completed, the library binaries are within the folder build/qbrobotics-driver and build/serial. They can be now used in your own application!

25 https://bitbucket.org/qbrobotics/qbdevice-api-7.x.x/src/production/
26 https://cmake.org/

CMake project setup

In order to set up a CMake project, using the compiled files, that implements your own application, we suggest to create a new project that has a structure similar to the following:

```
├── include
│   ├── qbrobotics_research_api
│   └── serial
├── libs
└── src
└── CMakeLists.txt
```

- the folder ***include/qbrobotics_research_api*** must contain the following header files:
  - **qbrobotics_research_api.h** - present in the folder *qbdevice-api-7.x.x/qbrobotics-driver/libs/research/include/qbrobotics_research_api* of qbdevice-api-7.x.x project.
  - **qbrobotics_research_commands.h** - present in the folder *qbdevice-api-7.x.x/qbrobotics-driver/libs/research/include/qbrobotics_research_api* of qbdevice-api-7.x.x project.
  - **qbsofthand2_research_api.h** - present in the folder *qbdevice-api-7.x.x/qbrobotics-driver/libs/research/include/qbrobotics_research_api* of qbdevice-api-7.x.x project.
- the folder ***include/serial*** must contain the following header file:
  - **serial.h** - present in the folder *qbdevice-api-7.x.x/serial/include*
- the folder **libs** must contain the compiled libraries obtained by following the instructions in the previous paragraph.
- the folder **src** must contain the main.cpp that exploits qbrobotics API functionalities.
- The CMakeLists.txt has to configure the project to link against the compiled libraries.

Once the folders setup is finished you can compile your own application by executing the commands used to compile the libraries in the previous folder.

An example of how to structure the project can be found in the Bitbucket repository API-test-SH2R-v7[27]. This project explains also how to handle the communication with *qb SoftHand2 Research* and how to control it.

The Research devices, manufactured by **qbrobotics**, are open source as the company's aim is to facilitate the work of our customers by allowing them to personalise the use of the device to their liking.
Within the source code, however, there are functions that if used improperly can damage the device and invalidate its warranty status.
What we ask is therefore that you only use the functions that you really need and make sure you fully understand how they work before you use them.
Our Support Team is always at your disposal for any clarification you may need, you can contact them at support@qbrobotics.com[28] address.

27 https://bitbucket.org/qbrobotics/api-test-sh2r-v7/src/master/
28 mailto:support@qbrobotics.com

6.3.4 Support, Bugs and Contribution

Since we are not only focused on this project it might happen that you encounter some trouble once in a while. Maybe we have just forget to think about your specific use case or we have not seen a terrible bug inside our code. In such a case, we are really sorry for the inconvenience and we will provide any support you need.

To help you in the best way we can, we are asking you to do the most suitable of the following steps:

1. It is the first time you are holding a *qb SoftHand2 Research* it is always a pleasure for us to solve your problems, but please consider first to read this guide carefully.
2. You are a beginner user stuck on something you completely don't know how to solve or you are experiencing unexpected behaviour: feel free to contact us at support@qbrobotics.com[29], you will receive the specific support you need as fast as we can handle it.
3. You are quite an expert user, everything has always worked fine, but now you have founded something strange and you don't know how to fix it: we will be glad if you open an Issue in the package of interest on our Bitbucket[30].
4. You are definitely an expert user, you have found a bug in our code and you have also correct it: it will be amazing if you open a Pull Request in the package of interest on our Bitbucket[31]; we will merge it as soon as possible.
5. You are comfortable with *qbrobotics* products but you are wondering whether is possible to add some additional software features: feel free to open respectively an Issue or a Pull Request in the package of interest on our Bitbucket[32], according to whether it is just an idea or you have already provided your solution.

In any case, thank you for using *qbrobotics*[33] solutions.

29 mailto:support@qbrobotics.com

30 https://bitbucket.org/qbrobotics/qbdevice-api-7.x.x/issues?status=new&status=open

31 https://bitbucket.org/qbrobotics/qbdevice-api-7.x.x/pull-requests/

32 https://bitbucket.org/account/user/qbrobotics/projects/ROS

33 https://www.qbrobotics.com/

7 Maintenance and warranty

7.1 General

Products of the company qbrobotics s.r.l. are produced using the most modern production methods and are subject of strict quality inspections. All information regarding our Warranty Policy can be found at this[34] site.

34 https://qbrobotics.com/technical-support/

8 FAQ and troubleshooting

For any issue not listed below, please contact our support team[35].

8.1 PC drivers and requirements

8.1.1 OS requirements

All the Long-Term Support (LTS) releases of the following OS that have not reached the End-Of-Life (EOL) term are officially supported.

Linux Ubuntu

Ubuntu 16.04, 18.04 and 20.04 are supported and recommended w.r.t. the other Linux distributions.

NOTE:

Note: different Linux distributions should also work fine, but they are not officially supported.

Microsoft Windows

As far as July 2020, Windows 10 version 1809+ and Windows 10 Enterprise 2015+ are supported and recommended.

8.1.2 Drivers installation

If you are using either Microsoft Windows or Apple macOS it is necessary to download and install the communication drivers from the FTDI VCP Drivers[36] page.

Linux Ubuntu

LINUX FRIENDLY:

You do not need additional drivers if you are running on any Linux distribution.

35 mailto:support@qbrobotics.com
36 https://www.ftdichip.com/Drivers/VCP.htm

However, if you have never set it up, you probably need to add your linux user to the `dialout` group to grant right access to the serial port resources. To do so, just open a terminal and execute the following command:

```
sudo gpasswd -a <user_name> dialout
```

where you need to replace the `<user_name>` with your current linux username.

NOTE:

Note: don't forget to logout or reboot.

Microsoft Windows

Install the driver for your operating system version to allow the serial port access through something like `` `COM7` ``.

WARNING:

Be aware that we support only serial port names between `` `COM1` `` and `` `COM9` ``.
Each time you connect a new device to your PC (or even a device to a new PC), you should check whether it has been set in the proper port range or not.
To change the serial port number:

1. go under `"Control Panel > Hardware and Sound > Device Manager"`;
2. explore the `"Ports (COM & LPT)"` branch, right-click on the displayed `` `COM` `` port and select `"Proprieties"`;
3. from there, select the `"Port Settings"` tab and click on the `"Advanced"` button `;`
4. lastly, choose from the drop-down menu a `` `COM` `` number between `` `COM1` `` and `` `COM9` `` and click on `"OK"`.

8.1.3 API installation requirements

Before using the qbAPI on your system, if not already installed it is good use to install a c++ compiler. Below here are listed all the necessary instructions for each operative system.

Linux Ubuntu

LINUX FRIENDLY:

You should have both `"gcc/g++"` and `"make"` already installed if you are running on any Linux distribution.

Apple macOS

Download `"XCode"` from the App Store, this will install `"gcc/g++"` and `"make"` utilities.

Microsoft Windows

Download MinGW[37] and install it. Open MinGW Installation Manager, from the left panel select basic setup, then from the right panel select mingw32-base and mingw32-gcc-g++, then click on `Installation -> Apply Changes`. This will install the *gcc/g++* compiler. To use it from the command line you need to provide to windows the binary path to the executable. Go to System Properties and click on Environment Variables. In the System Variable windows, look for `path`, select it and click `edit`. Go to the end of the Variable Value field, add a `;` separator and add the path to the binary folder for gcc (usually it is in `C:\MinGW\bin`).

Download the *make* utility from here[38]. Follow the installation instruction. In the end you will need to add the binary path to the Environment Variables. To do that follow the previous steps. (Usually the binary folder for the make utility is in `C:\Program Files (x86)\GnuWin32\bin`).

NOTE:

Note: if you have the CMD already opened when performing the installation, you probably will need to reopen a new one to be able to use the utilities.

8.2 Simplified CAD models of the qb SoftHand2 Research

8.2.1 Problem

Where can I find CAD models or simplified meshes of the qb SoftHand2 Research?

37 http://www.mingw.org
38 http://gnuwin32.sourceforge.net/packages/make.htm

8.2.2 Solution

We provides only simplified geometries of the qb SoftHand2 Research that you can find at our ROS description repository[39] (measures are expressed in millimeters).

As additional information, the phalanges are made of a polyamide 6 reinforced with 30% glass fiber (discontinuously placed), with the following main properties:

- density `: 1.36 [g/cm3]` ;
- tensile yield strength `: 165 [MN/m2]` ;
- tensile Young modulus `: 8500 [MN/m2]` ;
- percent elongation at break `: 3%` ;
- resilience (Izod method) `: 125 [J/m]` ;
- melting point `: 220 [°C]` ;
- Poisson ratio `: N/A` (for general polyamide 6, i.e. without glass fiber, it is approximately `0.4`).

8.3 The glove is deteriorated/broken

8.3.1 Problem

The glove is one of the component of the hand more prone to wear and tear.

The planned maintenance already makes up for this, but it may happen that the glove simply breaks during the normal usage.

WARNING:

Be aware that you must never grasp sharp objects, nor have them close to the robot during its operation, cf. safety warnings (see Page 3) .

A damaged glove can affect repeatability and grasping performance.

8.3.2 Solution

Unfortunately is not possible to ship you a whole new glove and let you proceed with its replacement.

Both the glove and the cover which secures it to the hand are crucial components of the hand and only assembling them inside our facilities guarantees the right setup and the advertised performance of grasping.

NOTE:

This is true also for the other elastic components inside the fingers.

[39] https://bitbucket.org/qbrobotics/qbhand-ros/src/dd75e2f70bbb470be4a2360177dbde6a7afbd638/qb_hand_description/meshes/?at=production-kinetic

In such a case you should raise a *"Service Request"* in our portal as soon as you notice the problem. We will do our best to ensure that this goes as quickly and as smoothly as possible.

8.4 The hand does not reopen correctly

8.4.1 Problem

The hand does not fully open up after setting the 0% position command: i.e. some of the fingers still remain a bit closed even if the motor position is correctly commanded to 0. Some of the fingers may move a bit slower than expected. Moreover, closure and reopening cycles present the same unexpected behavior at each open.

WARNING:

Be aware that this behavior can lead to hardware malfunctions if it is not solved quickly.
If it happens too often, try to grasp the objects more gently (e.g. smaller closure values) or use another approach strategy which produce a more natural grasp.

8.4.2 Solution

USER-FRIENDLY:

This problem can be easily solved by the user in a couple of minutes!

The thumb phalanges are not properly placed: indeed it can happen that they don't always go back to the original position (this highly depends on how fingers envelop the object and how much each phalanx dislocate from its original place).

As you can see in the video I attached below, you can try to manually adjust the thumb configuration by pulling or stretching each phalanx.

If you make sure that thumb phalanges are aligned, you should have no more troubles.

If the thumb seems ok, just try to do the same to all the other fingers: pull and twist a bit each distal phalanx until it seems to be in the proper place (set the hand reference to 0%, i.e. fully open, when you do such maintenance).

NOTE:

It is a good practice to perform these manual adjustments once in a while to ensure that the hand works at its best.

Please, let us know whether this solve the problem or you need further assistance.

8.5 The USB flash drive is not working

8.5.1 Problem

It is not so common, but it may happen that any of the files provided through the USB flash drive is corrupted.

8.5.2 Solution

You can find all the USB flash drive files in our download section at the bottom of the qb SoftHand2 Research page[40].

WARNING:

If any of the files is corrupted we recommend to format the flash drive and download again all the files from our website.

40 https://qbrobotics.com/product/qb-softhand-2-research/