Care-O-bot Manual

Manual for Care-O-bot users and administrators

Autors: Florian Weisshardt
Nadia Hammoudeh Garcia

Fraunhofer IPA

Institute for Manufacturing Engineering and Automation
Stuttgart, Germany

Last modified on Wednesday 23rd November, 2016

Contents

Nomenclature

X	robot number, e.g. X=3 for cob3-3
Y	pc number, e.g. Y=2 for pc2
IP	parameter that define the robot network 4.1

Chapter 1

Introduction

This manual is divided into three main parts. The first part (chapter 2) is intended for all users, it shows how to startup the robot, login and execute simple commands on the robot. The second part (chapter 4) addresses robot administrators and covers topics like setting up the pcs, configuring network and add new user accounts. The third part (chapter 5) offers some first help in case you need support.

You can always get the latest version of this manual at [1].

If you have comments, suggestions or would like to add something to the manual, please contact fmw@ipa.fhg.de.

[1] `https://github.com/ipa320/setup/blob/master/manual/Care-O-bot_manual.pdf`

Chapter 2

User Manual

NOTE: Before you start working with the robot you have to attend a safety introduction. If you haven't got one yet, please contact your local robot administrator.

The following user manual requires some basic knowledge about

- Linux/Ubuntu
- Source code management with git
- ROS usage

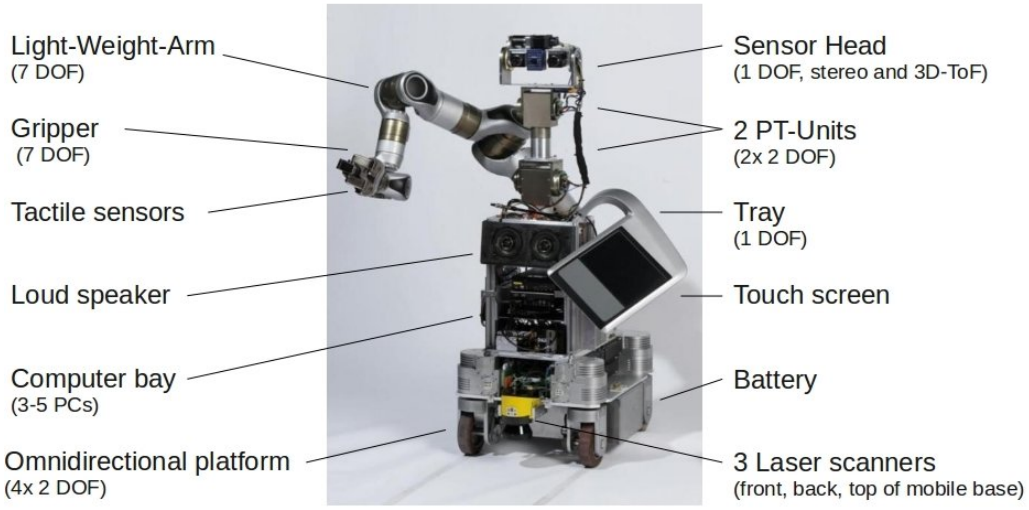
If you are missing some of this requirements or feel uncomfortable with what you are doing, please interrupt and ask somebody to help you before continuing.

2.1 Hardware overview

You can take a look of the technical data of Care-O-Bot on the official web site[1] and [2]. Also you can see the distribution of the different Care-O-bots at [3].

An overview of the robot hardware is shown in the following picture.

[1] http://www.care-o-bot-research.org/care-o-bot-3/technical-data
[2] http://www.care-o-bot-research.org/care-o-bot-3/components
[3] http://www.ros.org/wiki/Robots/Care-O-bot/distribution



2.2 Software overview

We defined a layer called *bringup layer*, which covers all hardware drivers and basic robot components. This is the layer where low level robot movements are enabled through the joystick or dashboard and the robot status including actuator and sensor information is acquired. In the following overview you can see the repositories which belong to the bringup layer.

- `cob_extern`: The `cob_extern` stack contains third party libraries needed for operating Care-O-bot. The packages are downloaded from the manufacturers website and not changed in any way.

- `cob_common`: The `cob_common` stack hosts common packages that are used within the Care-O-bot repository. Also the urdf description of the robot, which is the kinematic and dynamic model of the robot, 3D models of robot components, information required for gazebo to simulate the COB and utility packages or common message and service definitions.

- `schunk_modular_robotics`: This repository includes drivers and models for Schunk products, like powercubes or sdh.

- `cob_driver`: The cob_driver stack includes packages that provide access to the Care-O-bot hardware through ROS messages, services and actions. E.g. for mobile base, arm, camera sensors, laser scanners, etc...

- `cob_command_tools`: This stack provides the source code of the tools that you need to command the robot: `cob_command_gui`, `cob_dashboard`, `cob_script_server` and `cob_teleop`.
- `cob_robots`: The `cob_robots` stack collects Care-O-bot components that are used in bringing up a robot. The user's interface to the `cob_robots` stack is `cob_bringup`. In this package you find all the hardware configuration and launch files to bringup the hardware.
- `cob_environments`: This stack provides the parameters for default environment configurations.
- `cob_calibration_data`: This repository holds the current calibration data for Care-O-bot.

2.3 Batteries and power supply

Care-O-bot is powered by a 48V battery, which can be a Gaia rechargeable Li ion battery (60 Ah 48V) or a plumb battery. The batteries can be charged with a maximum of 56V at 10A. The robot can be plugged in during operation.

2.4 Emergency stop

Please be aware that not all robot movements are safe for you as an user, the robot itself and the environment. Therefore please don't hesitate to activate the emergency stop in situations which are not foreseen by the user. There are three ways of stopping the robot: On the robot you have two red buttons on the laterals, you can step into the safety fields of the Sick S300 scanners or you use the wireless emergency stop. Remember to recover the robot components with the command_gui after an emergency stop was activated, check the status of the components using the dashboard.

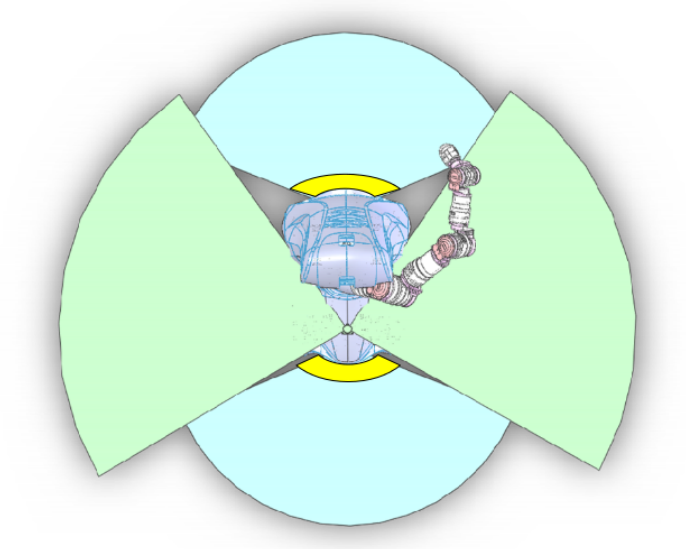
2.4.1 Emergency stop buttons

Press the red buttons on the left and right side of the torso. To release the emergency stop, turn the buttons so that they come out again. After that turn

the key to position II until you hear a "click".

2.4.2 Safety field from the Sick S300 scanners

If you step into the safety fields of the laser scanners the emergency stop is activated automatically. After the safety fields are free from any obstacle again the emergency stop is released on its own after a few seconds.



2.4.3 Remote emergency stop control

You can press the red button to stop the robot. To release the emergency stop you have to lift the red button and afterwards press the green button.

The remote emergency stop control can be disabled using the switch next to the key switch.

WARNING: This should usually not be used. It is only intended to be used if no remote control is available.

If the remote emergency stop control is disabled and the remote is within the wireless range, it should be placed in its base station. If you remove the remote from the base station, it might trigger an emergency stop.

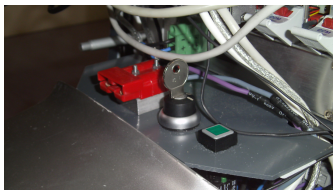
NOTE: If you hear a "click" while releasing the emergency stop, but the dashboard and command_gui still show that the emergency stop is activated, turn the key to position II again and hold it for some seconds until the dashboard or command_gui show no more emergency stop.

2.5 Running the robot

First you have to connect the power supply to the robot or you can use the battery pressing the battery button on the base. To switch the robot on you have to use the key. If you move it to position II and hold for a few seconds the robot will turn on. To turn of the robot turn the key to position I.

After starting up the robot the emergency stop circuit is still activated. To enable power for the motors, keep the safety area of the Sick S300 scanners free of any obstacle and release the emergency buttons. In the case that the wireless emergency stop is active, release it by releasing the red button and pressing the green button afterwards. You should hear a "click" as soon as the emergency stop is released.

For safety reasons if the robot is not supervised any more, e.g. during a break, the emergency stop has to be activated. Never operate the robot without a local person supervising it being aware of the safety instructions (see separate safety instructions).



2.5.1 Logging in to the robot pcs

For logging in with a remote PC to the robot pcs you have to have an account on the robot. If you don't have an account contact your local robot administrator to create one for you (see the section 4.1.5.1). Use ssh to login (in this example to pc1 of cob3-3)

```
ssh -X user_name@cob3-3-pc1
```

2.5.2 Bringup the robot

Note: The following steps can only be done once by one person at the same time. It is not possible to start for example mulitple roscore or bringup.

The first step to bringup the robot is to start a roscore, this is necessary to have communication between the nodes. You can run it using

```
roscore
```

Note: If your Care-O-bot has a KUKA LBR arm please see the section 2.7. Note: If your Care-O-bot has a Universal robot arm please see the section 2.8. If you want to run the robot you have a launch file for launching all the components of the robot. It is located in the package cob_bringup.

```
roslaunch cob_bringup robot.launch
```

Now all drivers and core components should be started so you can continue and have a look at the robot status in the dashboard or moving the robot using joystick or command_gui.

2.5.3 Using dashboard and command_gui

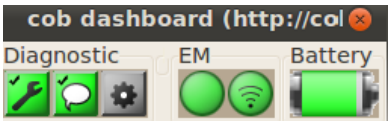
To know the state of all the components of the robot you can use the dashboard tool. To move the robot you can use the command_gui. You can start both with

```
roslaunch cob_bringup dashboard.launch
```

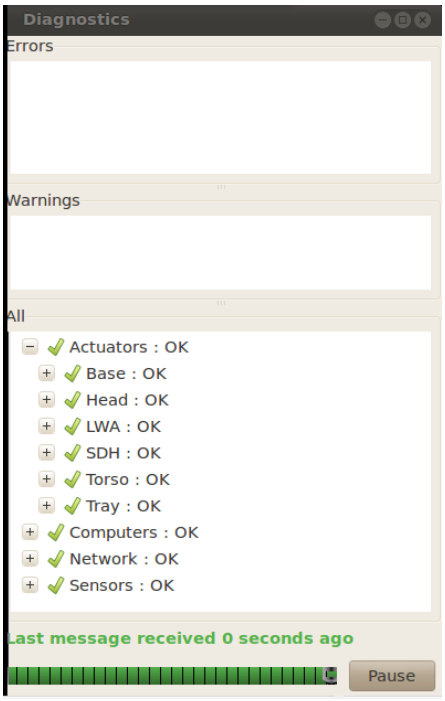
After launching this file you will see two GUIs on your screen: the smaller one is the dashboard, which gives you information about the current status of the robot and its components as well as the emergency stop status. The bigger one is called command_gui and offers a wide range of buttons to move the robot to predefined positions using low level control commands. It also offers buttons for initialising and recovering the actuators. The emergency stop status is visible in the upper left corner of the command_gui. Before initialising or moving the robot, check that the status is OK.

2.5.3.1 cob_dashboard

The dashboard is an important tool where you can check the state of the robot, it is recommended that you have it always opened. The dashboard looks like this:



If you click the first button, you will see a new window popping up with three levels: Errors, Warnings and All. There you can see the state of each component at any time. The status monitoring is divided into Actuators, Sensors and other. The other buttons are for showing diagnostics, motors, emergency status and battery state. In the case of the Care-O-bot we have disable the buttons for the Motors, you see them always in red.



All diagnostics information for the actors and sensors should be green after successfully initializing the components, see section 2.5.3.2.

2.5.3.2 cob_command_gui

The command gui can be used for sending low level movement commands to the robot components. The standard view of the command_gui is 2.1

In this screen-shot 2.1 you can see different columns: general, base, torso, tray, arm settings, arm pos, arm traj, sdh and eyes. The first column is very important, when you run the robot. If you want to move it, first initialize all components with a click on `init all` and make sure that the **emergency stop is not active**. It will take some time and the actuators will do their homing sequence if necessary. After an emergency stop was activated, each component has to be recovered. Therefore press the button `recover all`.

The other columns of the components have different predefined positions where you can move to. Additionally Each component has a `stop`, `init` and `recover` button, to stop, initialize or recover a single component.

Figure 2.1: Standard view of the command_gui

2.5.4 Rviz

RVIZ is a tool that visualizes data from the robot, e.g. the sensor data from the laser scanners, but also information about the coordinate systems and transformations or the images from the cameras. You can add your own items to RVIZ to visualize topics, see more information at [4].

RVIZ needs to be started on your local machine. To be able to visualize topics from the robot export your `ROS_MASTER_URI` to the robot

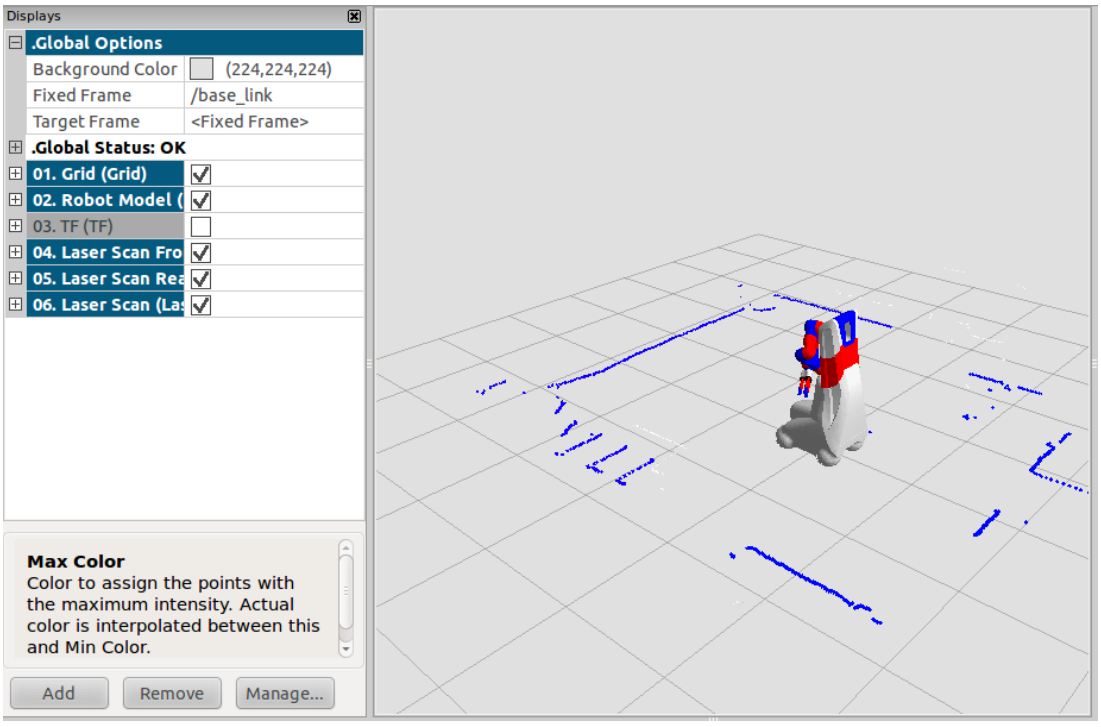
```
export ROS_MASTER_URI=http://cob3-X-pc1:11311
rosrun rviz rviz
```

To use a predefined configuration for Care-O-bot start rviz with

```
export ROS_MASTER_URI=http://cob3-X-pc1:11311
roslaunch cob_bringup rviz.launch
```

You will see a screen like this:

[4] `http://www.ros.org/wiki/rviz`



2.5.5 Joystick

To be able to use the joystick, initialize the components using the command_gui. For moving the robot components, the dead-man_button has to be pressed all the time, as soon as the button is released all hardware components will be stopped immediately.

- For moving the base: Hold the dead-man button and use the base rotation and translation axis to move the base.
  There is a slip button locates on the backside of the joystick(it doesn't show on the picture), before moving the robot, slip it to the "D" position.

Have a look at the following image to see which buttons command which components.

2.6 Power down the robot

To power down the robot press the emergency button and turn the key to position I, disconnect the power cable and turn off the power supply.

2.7 Operating KUKA LBR

2.7.1 Initializing the LBR

In order to initialize the LBR, login to pc1 with:

```
ssh -X user_name@cob3-X-pc1
```

and from pc1 login to the lbr box via telnet

```
telnet lbr_box
```

Now you can initialize the LBR, therefore type

```
<1
```

in the telnet terminal and wait for the line **Stable state RUNNING reached!**. If you dont see this line, contact your robot administrator: theres possibly something wrong. For a correct operation of the LBR it is important that the controller gets to know both states, emergency stop active and released. Therefore activate the emergency stop for about 15 seconds until you see **administrationThread emerg change 1 0**. After releasing the emergency stop again you should see **administrationThread emerg change 0 1**. To lougout from telnet use Ctrl+D or type

```
logout
```

With that procedure the LBR is ready to be used.

2.7.2 Starting the LBR

You can bringup the whole robot

```
roslaunch cob_bringup robot.launch
```

or only the LBR using:

```
roslaunch cob_bringup lbr.launch
```

The LBR is now ready to be moved. This can be done using the cob_command_gui or the joystick.

!! ATTENTION !!

The operation and recovering of the LBR after an emergency stop has to be done carefully following the steps below, Unexpected and fast movements can happen which could damage you, the robot and the environment.

- First of all bring the robot away from any obstacles, especially obstacles in the workspace of the LBR.

- Press the LBR hardware reset button shown in figure 2.7.2 and wait for 30 sec.

The next steps depend on how the emergency stop got activated. There are two versions:

- (a) is a short version which is not 100
- (b) is a safe version which involves some more steps

2.7.2.1 Version (a), the short one

can be used if the emergency stop was activated while the LBR was not moving and not stressed with low stiffness values. If you can use version (a), you can simply launch the LBR node again with

```
roslaunch cob_bringup lbr.launch
```

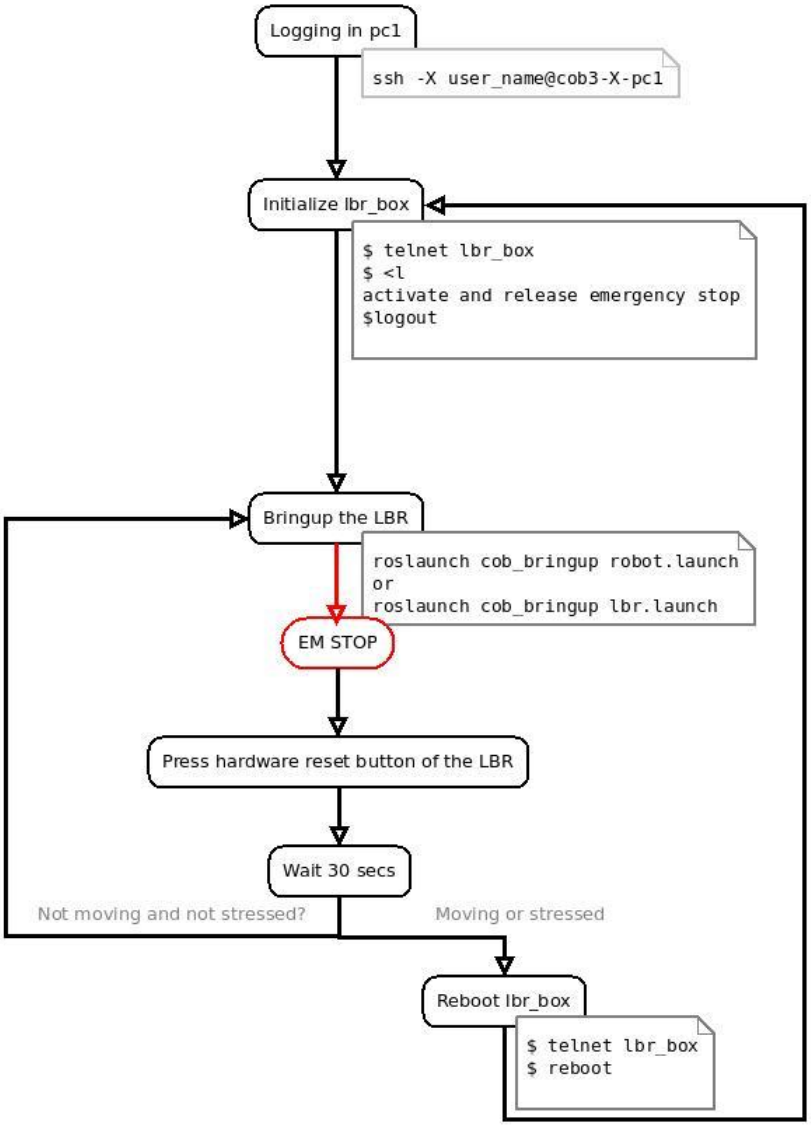
2.7.2.2 Version (b), the safe one

If the LBR was moving or stressed with low stiffness values use version (b). If you are not sure how the emergency stop got activated, use the safe version (b). If you have to use version (b) you will have to reboot the lbr box with

```
telnet lbr_box
reboot
```

Wait for the lbr box to reboot and start again with the initializing procedure from section 2.7.1

2.7.2.3 LBR handling overview



2.8 Operating Universal robot

2.8.0.4 Start-up the UR controller

You can turn on the UR controller by pressing the button next to the key. After pressing the button the button should light up in green and the UR controller should boot up.

Next, you will need to release the emergency stop and initialize the UR arm by following the user interface on the touch panel.

NOTE: You can only release the emergency stop if the UR controller is booted up.

2.8.0.5 Operating the arm

For operating the arm a ROS node needs to be started. This is done by the bringup launch file

```
roslaunch cob_bringup robot.launch
```

or separately with

```
roslaunch cob_bringup ur_solo.launch ur_ip:=<<IP ADRESS OF YOUR UR
    CONTROLLER>>
```

After that you can directly operate the arm by using the command_gui or send a FollowJointTrajectoryAction to the arm.

2.8.1 The UR connector

To extend the workspace there's a ur connector which is an external 7th axis to the arm to be able to operate on the front and back side.

2.8.1.1 Start-up the UR connector

The ur_connector should be powered and ready to operate once there is no emergency stop active.

2.8.1.2 Operating the UR connector

For operating the ur_connector a ROS node needs to be started. This is also done by the bringup launch file, which if started before should no be relaunched.

```
roslaunch cob_bringup robot.launch
```

or separately with

```
roslaunch cob_bringup ur_connector_solo.launch ur_ip:=<<IP ADRESS
   OF YOUR UR CONTROLLER>>
```

After that you can directly operate the ur_connector by using the command_gui or send a FollowJointTrajectoryAction to the arm.

Chapter 3

Packing the robot

In this section, the necessary steps for preparing the robot for shipment are described. Please consult the photos in this section for details. A checklist of the material that you should take along in the robot shipping box in order to properly pack the robot is listed at [1]. Moreover, you should also always travel with a "first-aid" box that allows you to address common problems that may arise with the robot. The checklist for this first-aid box is listed at [2].

The necessary packing steps are as follows:

- *Prepare the box.* Take out everything that could be in the way. Lay the tension belt flat on the ramp or under it. Open the front door (snap-on magnet). Remove the **2** rear cover from the robot.

- *Drive the robot into the box.* Take care that the robot is centered in the box as much as possible. Make sure that the black marks are free and there are some centimeters to the board of the lower belts.

- *Push emergency stop button, then power off.* **Attention:** Always push the emergency button before turning power off. Otherwise, the torso may lose its reference position, requiring a lenghty recalibration process.

- *Fasten the robot with a tension belt, but not too hard!* **Attention:** Make sure the fastener or the tension belt is not in contact with the robot cover.

[1] `https://raw.github.com/ipa320/setup/master/manual_ipa/ShippingBox.pdf`
[2] `https://raw.github.com/ipa320/setup/master/manual_ipa/FirstAidKit.pdf`

- *Use foam material to protect the robot and to prevent slipping.* Make sure that no part of the robot is in contact with the tension belts and the robot hast a good safty distance all around to the box.
- *Air travle* For air travel, fix an additional tension belt around the box. Moreover, seal the hinges with a plastic seal.

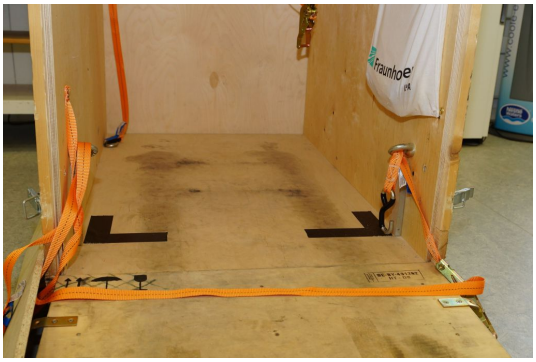



Figure 3.1: Step 0: Remove board and put the belt aside (left). Open the front door (right).

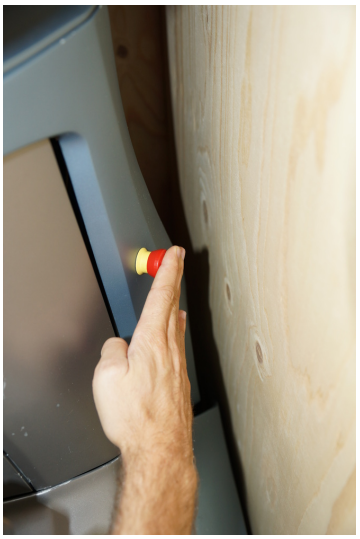



Figure 3.2: Step 1: Drive the robot in the box and park it as centrally as possible (left). **Always push the emergency stop button before power off!**

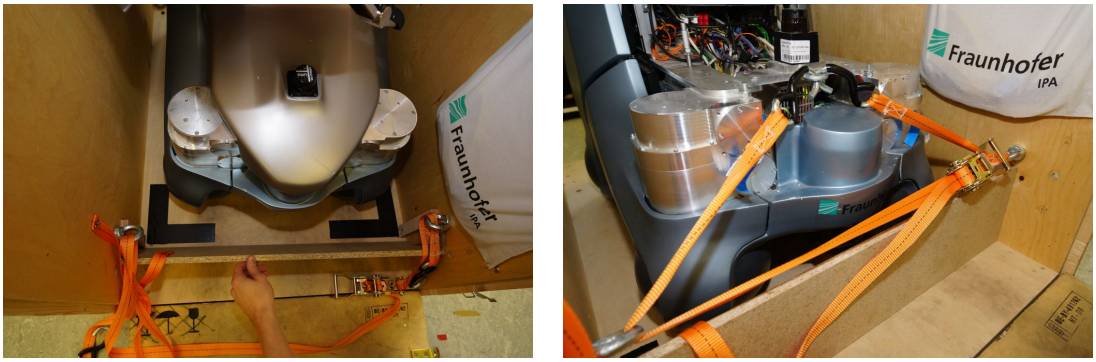


Figure 3.3: Step 2: Place the borad under the lower belts (left). Hook in the ends of the belt and fasten it **moderatly (right).**

Figure 3.4: Step 3: Hook in the ends of the upper belt (left). Fasten the belt from the front (right).

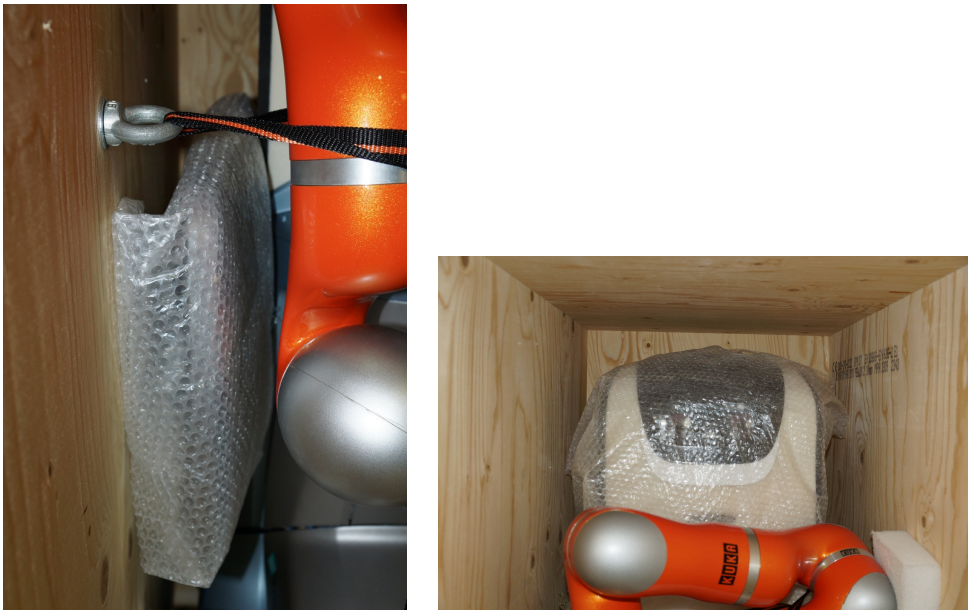


Figure 3.5: Step 4: Put bubble wrap around the tray and over the head.

Figure 3.6: Step 5: Put the "first-aid" box in holder and fasten the strap (left). Wrap the rear covers in bubble foil an put it infront of the robot. (right).

Figure 3.7: Step 6: Pack the accessory box as shown (left) and place it in the rails (right).

Figure 3.8: For air travel, also fix an additional tension belt around the box (left), and seal the hinges with a plastic seal (right).

Chapter 4

Administrator manual

The following administrator manual requires some enhanced knowledge about

- Linux/Ubuntu
- Network configuration
- Source code management with git
- ROS installation and usage

If you are missing some of this requirements or feel uncomfortable with what you are doing, please interrupt and ask somebody to help you before continuing.

4.1 Setup robot pcs

On all Care-O-bots there are at least two pcs. Some Care-O-bots have an optional third pc, which is not covered by this manual. Within this section we will guide you through setting up new pcs. When nothing otherwise is mentioned the following instructions are for both pc1 and pc2, please do the same steps on both pcs.

To pc1 all actuators are connected, sensors are connected both, to pc1 and pc2. All camera sensors are connected to pc2, whereas all other sensors like e.g. laser scanners are connected to pc1. By default pc3 is not connected to any hardware and therefore can be used as additional computing power.

4.1.1 Install operating system

The first step is to install the operating system for each pc, which means pc1 and pc2 (optionally pc3). We are using Ubuntu as the main operating system for the robot. We recommend to install the **Ubuntu 14.04 LTS (long term stable) 64-bit** version because this version is well tested to work with the hardware.

First please install Ubuntu (English version) creating a normal swap partition. Please choose `robot` as an admin account with a really safe password which should only be known to the local robot administrator. The hostname of the pc should be `cob3-X-pc1` and `cob3-X-pc2`.

4.1.2 Setup internal robot network

Inside the robot there's a router which connects the pcs and acts as gateway to the building network. Setup the router with the following configuration.

The ip address of the router should be `192.168.IP.1` and for the internal network dhcp should be activated. Use `cob3-X` as hostname for the router. Register the MAC addresses of pc1 and pc2 so that they get a fixed ip address over dhcp. Use `192.168.IP.101` for pc1 and `192.168.IP.102` for pc2. Enable portforwarding for port `2201` to `192.168.IP.101` and for port `2202` to `192.168.IP.102`, where IP parameter is defined depending on the robot 4.1.

cob3-X	IP=X
raw3-X	IP=40+X
desire	IP=100

Table 4.1: IP address definition

After ensuring that the network configuration of the router is setup correctly, we can configure the pcs. All pcs should have two Ethernet ports. The upper one should be connected to the internal router.

4.1.3 Install basic setup

Next we have to install some basic tools for the further setup of the pcs. In order to install the packages a internet connection is needed.

The full Care-O-bot installation can be done using a bash script. The script is in the setup repository, get it using the following command:

```
wget https://raw.githubusercontent.com/ipa320/setup/master/
   InstallCob3.sh
chmod +x InstallCob.sh
```

The installation script needs the parameters robot, ip address and installation mode, where:

- -r robot: is the robot name (cob3-X)
- -ip :ip address for your actual installation pc, use `192.168.IP.101` for pc1 and `192.168.IP.102` for pc2, if you have a third pc, assign it the ip `192.168.IP.103`
- -m installation mode: on Care-O-bot there are two different types of computers, the master pc and the slave. The master PC has a large hard disk , and works as a NFS system server, the other computers will be the clients

The script allow different types of installation:

1. `Basic Installation` It is composed by the following steps:
   - Install basic tools (vim, meld, terminator ...)
   - Install and configure openssh
   - Allow robot user to execute sudo command without password
   - Setup root user (in this step the user will be asked for a password)
   - Install ROS
   - Setup udev rules
   - Setup bash environment
2. `Setup NTP and NFS` This option will configure the NFS system depending on the installation mode, it is important that the master pc is already installed and per network reachable before install the slave computers, otherwise the installation process will be cancelled. After this installation it is necessary restart the computer.

3. `Full Installation` A full installation means the combination of 1 and 2 (Basic Installation + Setup NTP and NFS)

4. `Cob setup` This step holds the recommended configuration of the robot home directory. This step can only be execute after a full installation.

After the network is configured properly we can setup a NFS between the robot pcs. pc2 will act as the NFS server and pc1 as NFS client, start always the installation with the master pc, in this case `cob3-X-pc2`

Run the Install script on pc2:

```
./InstallCob3.sh -r cob3-X -ip 192.168.X.102 -m master
```

Please take your time to check that the given parameters are right before continue, after the verification choose the option 3 (Full installation). The installation will take around 30 minutes , depending on the internet connection. During the process the user will be asked to confirm and verify the selected options. There will be one question:"Do you agree with above patent declaration?", you should say "no".

After the successfully pc2 installation and with all the computers connected to the router, repeat the process with the client pcs (cob3-X-pc1):

```
./InstallCob3.sh -r cob3-X -ip 192.168.X.101 -m slave
```

After the full installation of all the Care-O-bot computers, the home directory can be configured calling the Install script again on master pc (cob3-X-pc2): Run the Install script on pc2:

```
./InstallCob3.sh -r cob3-X -ip 192.168.X.102 -m master
```

And choose the option 4 (Cob setup).
There are several errors you might meet:

- "/u/robot/git/care-o-bot/src/src workspace doesn't exist": add "cd /u/robot/git/care-o-bot" before "catkin_make install".
- "ssh root@cob3-6-pc1: connection refused": open /etc/ssh/sshd_config file, make sure this is correct: "PermitRootLogin yes".

4.1.4 Setup hardware components

In order to use the different hardware components we have to install the drivers and set permission rights. All hardware configuration is stored in the `cob_hardware_config` package.

Setup can bus drivers This step is necessary for all drivers with can bus interface (Schunk powercubes, Schunk sdh, head axis, base). In general both can drivers from Peak Systems[1] `libpcan` and ESD[2] `libntcan` can be used. Installation instructions can be found in the package documentation of libpcan `http://www.ros.org/wiki/libpcan` and libntcan `http://www.ros.org/wiki/libntcan`.
The installation can be done using a bash script in the `cob_extern` package, you can clone it to your workspace use:

```
git clone https://github.com/ipa320/cob_extern.git
```

If you got some errors, you can check the possible errors posted on: `http://wiki.ros.org/allegro_hand_ros`. If you still can't fix the error, try to install it without the script:

- change to pc1, download "peak-linux-driver-8.2" package from: `http://www.peak-system.com/fileadmin/media/linux/index.htm`, you can also download the Manual PDF as a reference.
- get into the "peak-linux-driver-8.2" package, build binaries and install package:

```
make clean
make NET=NO_NETDEV_SUPPORT
sudo make install
```

- load Driver:

```
cat /proc/pcan
sudo modprobe peak_pci
sudo modprobe peak_usb
```

[1] `http://www.peak-system.com/`
[2] `http://www.esd-electronics.com`

- restart pc.

4.1.4.1 Scanners

Sick S300 laser scanners The Sick S300 scanners on the front side and backside of the robot are connected via USB to pc1. Configuration is done in the `cob_hardware_config` package in `config/laser_front.yaml` and `config/laser_rear.yaml`, documentation about parameters in [3].

To receive data from the Sick S300 scanners check if the user is in the `dialout` group

```
groups
```

For testing you can run the front laser scanner with

```
roslaunch cob_bringup sick_s300.launch name:=front
```

To check if there is some data published use

```
rostopic hz /scan_front
```

Check the rear scanner in the same way, the argument name is rear (`name:=rear`) and the topic `/scan_rear`.

You have to configurate the scanners and setup the safety region for each scanner. It can be done using the Sick CDS software [4].

Note on new firmware (2.10 and greater): Sick changed their scanner product line. From now on (April 2013) the Sick Professional CMS, that we used so far, has been replaced by the Sick Expert. The main difference is, that there is now no additional CMS version anymore. Expert equals to CMS, whereas Professional no longer has the CMS capabilities. The new scanners are also shipped with a new firmware (2.10). It is thus not possible anymore, to simply flash an old configuration as it might break the scanner. Additionally, the new scanners also have a new telegram protocol version, which is not compatible with the current driver. However, it can be configured to use the old protocol versions. How to setup the scanner with the new firmware is described in the following.

[3] `http://www.ros.org/wiki/cob_sick_s300`

[4] `http://www.sick.com/group/EN/home/service/software/Pages/configurationsoftwarecds.aspx`

How to setup Sick laser scanners with firmware version 2.10: Make sure you have CDS 3.6.7 installed (even though it is not the latest one, this is the recommended version from the Sick support). When the scanner is connected and the CDS has started, add a new device to the respective COM port (do NOT hit the `Recognize Project` button). Choose the correct device: S300 Expert (CMS should be entered automatically). Check the `Compatibility Mode` box; this serves to enable the old telegram protocoll version. After that you can edit the configuration or import one (the current configurations for the scanners can be found in the robot specific folders on GitHub[5], `raw3-3` contains configurations for the new scanners). If you are not sure which scanner your robot has, have a look at the distribution site on ros.org[6].

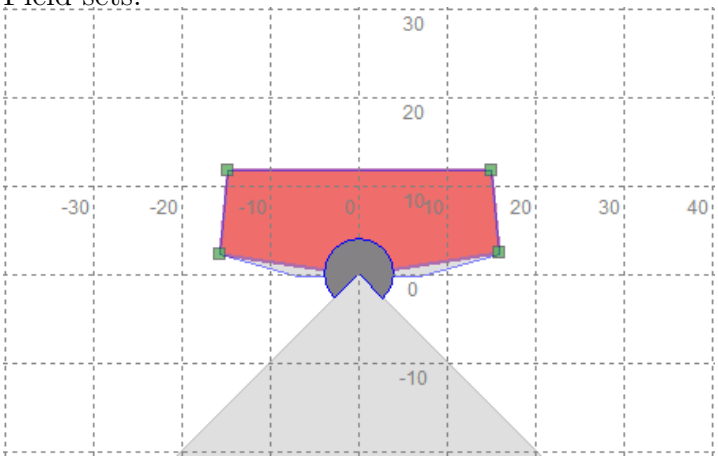
Note on importing scanner fields: Please note that even though you can easily export and import any configurations, the scanner fields are not contained in the configuration files. You need to export and import the scanner fields separately using the CDS. This can be done in the `File` menu.

We recommend the following configuration:

- System parameters:
  - Aplication name: COB3
  - Device name S300[H]: S300-V
  - Device name S300[G]: S300-H
  - Name of the user: IPA
- Resolution/scanning range:
  - Aplication: Mobile
  - Resolution: 30 mm
- Restart:
  - Time delayed by: 2 Seconds

[5] `https://github.com/ipa320/setup/tree/master/backup_configuration`
[6] `http://www.ros.org/wiki/Robots/Care-O-bot/distribution`

- Field sets:



  The safety region should have 4 points:

  - x=15 cm y=12 cm
  - x=-15 cm y=12 cm
  - x≈15 cm y≈2.5 cm (depend on the cover)
  - x≈-15 cm y≈2.5 cm (depend on the cover)

- Measure data output:

  - Baud rate: 500 kBaud
  - Send mode: Continuous data output
  - Measured data output: Distance
  - Beginning: -45
  - End: 225

Hokuyo URG laser scanner The Hokuyo laser scanner is connected to PC1 via USB. The configuration of this device is defined in the launch file as parameters `cob_bringup/components/laser_top.launch`.

For testing the scanner you have to call the run the node:

```
rostlaunch cob_bringup hokuyo.launch name:=top
```

You can check the output of the topic:

```
rostopic hz /scan_top
```

4.1.4.2 Base

Relayboard The Relayboard is connected to PC1 via USB. The configuration file (`relayboard.yaml`) is in the package `cob_hardware_config` inside the folder `cob3-X/config`.

For testing the Relayboard you have to launch the file:

```
roslaunch cob_bringup relayboard.launch
```

And checking the output of the topic `/emergency_stop_state` prove its correct operation:

```
rostopic echo /emergency_stop_state
```

Base You have to configure the elmo controllers, all the information about the driver is in [7], you find the parameters of this configuration in `cob_hardware_config` package in `cob3-X/config/base`.

For testing you can launch the base with:

```
roslaunch cob_bringup base_solo.launch
```

Init the component using the service:

```
rosservice call /base_controller/init
```

And try to move the base using the joystick 2.5.5

4.1.4.3 Torso

Torso with PRL modules There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the powercubes and the other one is done through ROS parameters.

First make sure that the powercube settings in the firmware are correct. You can see and modify the parameters using the windows based PowerCubeCtrl software

[7] `http://www.ros.org/wiki/cob_base_drive_chain`

from the schunk homepage[8]. A sample configuration can be fount at [9].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/torso.yaml`. Documentation about the ROS parameters can be found at [10] and [11].

There is a launch file in cob_bringup per driver for testing:

```
roslaunch cob_bringup component_solo_basics.launch
roslaunch cob_bringup schunk_powercube_chain_driver.launch
   component_name:=torso
```

For initialing and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

Torso with PRL-plus modules There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the MTS and the other one is done through ROS parameters.

First make sure that the MTS settings in the firmware are correct. You can see and modify the parameters using the windows based MTS software from the schunk homepage [12]. A sample configuration can be fount at [13].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/torso_controller.yaml`, `cob3-X/config/torso_driver.yaml` and `cob3-X/config/canX.yaml`. Documentation about the ROS parameters can be found at [14].

There is a launch file in cob_bringup per driver for testing:

```
roslaunch cob_bringup component_solo_canopen.launch component_name
   :=torso can_module:=canX
```

[8] `http://www.schunk-modular-robotics.com/left-navigation/service-robotics/service-download/software/prl/powercube.html`

[9] `tbd`

[10] `http://www.ros.org/wiki/schunk_powercube_chain`

[11] `http://www.ros.org/wiki/cob_trajectory_controller`

[12] `http://www.de.schunk.com/schunk/schunk_websites/service/download_tracking.html?href=MTS_v_1_56_20130904.zip&country=DEU&lngCode=DE&lngCode2=DE`

[13] `tbd`

[14] `http://wiki.ros.org/ros_canopen`

For initiating and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

4.1.4.4 Trays

One DOF tray with PRL modules There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the Powercubes and the other one is done through ROS parameters.

First make sure that the powercube settings in the firmware are correct. You can see and modify the parameters using the windows based PowerCubeCtrl software from the schunk homepage[15]. A sample configuration can be fount at [16].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/tray.yaml`. Documentation about the ROS parameters can be found at [17] and [18].

There is a launch file in cob_bringup per Schunk component for testing:

```
roslaunch cob_bringup component_solo_basics.launch
roslaunch cob_bringup schunk_powercube_chain_driver.launch
   component_name:=tray
```

For initialing and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

3 DOF tray and one DOF tray with PRL-plus modules There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the MTS and the other one is done through ROS parameters.

First make sure that the MTS settings in the firmware are correct. You can see and modify the parameters using the windows based MTS software from the

[15] `http://www.schunk-modular-robotics.com/left-navigation/service-robotics/service-download/software/prl/powercube.html`

[16] `tbd`

[17] `http://www.ros.org/wiki/schunk_powercube_chain`

[18] `http://www.ros.org/wiki/cob_trajectory_controller`

schunk homepage[19]. A sample configuration can be fount at [20].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/tray_controller.yaml`, `cob3-X/config/tray_driver.yaml` and `cob3-X/config/canX.yaml`. Documentation about the ROS parameters can be found at [21].

There is a launch file in cob_bringup per driver for testing:

```
roslaunch cob_bringup component_solo_canopen.launch component_name
   :=tray can_module:=canX
```

For initiating and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

4.1.4.5 Arms

Schunk powercubes LWA There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the powercubes and the other one is done through ROS parameters.

First make sure that the powercube settings in the firmware are correct. You can see and modify the parameters using the windows based PowerCubeCtrl software from the schunk homepage[22]. A sample configuration can be fount at [23].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/lwa.yaml` or `/config/torso.yaml`. Documentation about the ROS parameters can be found at [24] and [25].

There is a launch file in cob_bringup per schunk component for testing:

```
roslaunch cob_bringup lwa_solo.launch
```

[19] `http://www.de.schunk.com/schunk/schunk_websites/service/download_tracking.html?href=MTS_v_1_56_20130904.zip&country=DEU&lngCode=DE&lngCode2=DE`

[20] `tbd`

[21] `http://wiki.ros.org/ros_canopen`

[22] `http://www.schunk-modular-robotics.com/left-navigation/service-robotics/service-download/software/prl/powercube.html`

[23] `tbd`

[24] `http://www.ros.org/wiki/schunk_powercube_chain`

[25] `http://www.ros.org/wiki/cob_trajectory_controller`

For initialing and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

Schunk powercubes LWA4D There are two parts where configuration needs to be set. One part of the configuration is done inside the firmware of the MTS and the other one is done through ROS parameters.

First make sure that the MTS settings in the firmware are correct. You can see and modify the parameters using the windows based MTS software from the schunk homepage[26]. A sample configuration can be fount at [27].

The ROS configuration is done in `cob_hardware_config`, in `cob3-X/config/arm_controller.yaml`, `cob3-X/config/arm_driver.yaml` and `cob3-X/config/canX.yaml`. Documentation about the ROS parameters can be found at [28].

There is a launch file in cob_bringup per driver for testing, for the canopen the can port where it is connected has to be given as argument:

```
roslaunch cob_bringup component_solo_canopen.launch component_name
   :=arm can_module:=canX
```

For initiating and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

KUKA LBR The Kuka LBR PC is connected to PC1 by Ethernet cable, you have to configure the IP address adding to the file /etc/network/interfaces the following lines:

```
auto eth1
iface eth1 inet static
address 192.168.42.15
netmask 255.255.255.0
```

[26] `http://www.de.schunk.com/schunk/schunk_websites/service/download_tracking.html?href=MTS_v_1_56_20130904.zip&country=DEU&lngCode=DE&lngCode2=DE`

[27] `tbd`

[28] `http://wiki.ros.org/ros_canopen`

You can also give the alias name (lbr_box) to the arm ip address, adding the this line to /etc/hosts:

```
192.168.42.146 lbr_box
```

You have to compile the cob_lbr repository (this repository is not public, please ask us for it) :

```
rosmake cob_lbr
```

For testing , the first step is check the connection with the arm PC:

```
ping 192.168.42.146
ping lbr_box
```

The initialization process for LBR is done through the lbr_box, you can login from PC1 to the lbr_box via telnet:

```
telnet lbr_box
```

Now you can initialize the LBR, therefore type:

```
<l
```

in the telnet terminal and wait for the line Stable state RUNNING reached!. If you dont see this line, contact your robot administrator: theres possibly something wrong. For a correct operation of the LBR it is important that the controller gets to know both states, emergency stop active and released. Therefore activate the emergency stop for about 15 seconds until you see administrationThread emerg change 1 0. After releasing the emergency stop again you should see administrationThread emerg change 0 1. To lougout from telnet use Ctrl+D.

Now you can bringup the LBR with the command:

```
roslaunch cob_bringup lbr.launch
```

For moving you can use the cob_command_gui 2.5.3.2.

Universal Robot 5 The arm with universal robot manipulator has 2 components the universal robot and a connector. This connector is managed by ipa_canopen package **??**.

TODO: configure the module with the windows software TODO: config files (cob_hardware_config) ur_connector_canopenmaster.yaml and ur_connector.yaml

There is a launch file in cob_bringup for launching the driver:

```
roslaunch cob_bringup ur_connector_canopen_solo.launch
```

For initialing and moving you can use the cob_command_gui 2.5.3.2.

These components should be manual calibrated as it is explained on 4.3.1.

4.1.4.6 Gripper

Schunk SDH with tactile sensors There is only one specific Firmware and library version supported. The version number is listed on [29]. Add the robot user to the group `dialout` in the `/etc/group` file.

You can launch the SDH with the following file:

```
roslaunch cob_bringup sdh_solo.launch
```

You can init and move the SDH using the cob_command_gui (2.5.3.2)

4.1.4.7 Head

Head axis The configuration is in the package `cob_hardware_config` in `cob3-X/config/head.yaml`. Documentation about parameters can be found at [30].

For testing the head axis you can launch head_solo.launch:

```
roslaunch cob_bringup head_solo.launch
```

This component needs a manual calibration, for more information please see the section 4.3.1.

[29] `http://www.ros.org/wiki/schunk_sdh`
[30] `http://www.ros.org/wiki/cob_head_axis`

4.1.4.8 Sensors

Prosilica cameras The IP address for the cameras should be `192.168.21.101` for the right camera and `192.168.21.102` for the left camera. You can set the camera IP address using the instructions on [31].

Sometimes the graphical network manager causes troubles, so it is best to remove it

```
sudo apt-get remove network-manager
```

After removing the network manager we will have to edit `/etc/network/interfaces` manually, you can do it copying the following lines on the pc2:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 192.168.21.99 # ip adress for camera network
netmask 255.255.255.0 # netmask
```

Tray sensors The tray sensors are defined in udev_rules and the library libphidgets[32] for this component is already built in the bringup level.

Kinect The kinect is connected to pc2. For testing you have to launch the kinect driver on pc2.
On pc1:

```
roscore
```

On pc2:

[31] `http://www.ros.org/wiki/cob_camera_sensors/AVT_Prosilica`
[32] `http://www.ros.org/wiki/libphidgets`

```
roslaunch cob_bringup openni2.launch
```

You can check the image in a remote pc:

```
export ROS_MASTER_URI=http://cob3-X-pc1:11311
rosrun image_view image_view image:=/cam3d/rgb/image_color
```

And the point cloud checking the topic:

```
export ROS_MASTER_URI=http://cob3-X-pc1:11311
rostopic hz /cam3d/depth/points
```

Or use the tool rviz as it is explained on section 2.5.4.

4.1.4.9 Touch Screen

4.1.4.10 Sound

4.1.5 Create new user accounts

4.1.5.1 Preparation

Due to the fact that all users need to be in the correct user groups, that the bash environment needs to be setup correctly and that user ids need to be synchronised between all pcs for the NFS to work, we facilitate the creation of a new user with a `cobadduser` script. Before you add the first user using the `cobadduser` script, please uncomment the following two lines in `/etc/adduser.conf` on pc2

```
EXTRA_GROUPS="dialout cdrom floppy audio video plugdev users"
ADD_EXTRA_GROUPS=1
```

To synchronize the accounts and passwords we need remote root access to all pcs. Therefore make sure you did the steps from sec **??**.

4.1.5.2 Account creation

After finishing the preparation step you can add new users. On pc2 and with administration rights you can use the following instruction

```
cobadduser new_user_name
```

An ssh-key and passwordless login to all robot pcs should work now. You can test it with

```
ssh cob3-X-pc1
ssh cob3-X-pc2
```

Logging in from one to another robot pc should work without password.

4.2 Network infrastructure for external access to the robot

For the robot internal network setup please refer to section 4.1.2.

Make sure you have name resolution and access to the robot pcs from your external pc. To satisfy the ROS communication you need a full DNS and reverse DNS name lockup for all machines. Check it from your remote pc with

```
ping 192.168.IP.101
ping cob3-X-pc1
```

and the other way round try to ping your remote pc from one of the robot pcs

```
ping your_ip_adress
ping your_hostname
```

If ping and DNS is not setup correctly, there are multiple ways to enable access and name resolution.

4.3 Calibration

The calibration is divided into two steps, the manual preparation and automatic calibration. All calibration parameters are defined in `cob_calibration_data/cob3-X/calibration/calibration.urdf.xacro`

4.3.1 Manual calibration

Components with hardware reference positions Move the components to their home position using the build in hardware reference positions, e.g. nonius of Kuka LBR.

Components without hardware reference positions You will have to manually calibrate the reference positions for each joint, e.g. for all Schunk powercubes in lwa, torso and tray as well as the head_axis with a water balance. Therefore move the component into its home position, which means that the joint values for that configuration should all be zero. Record e.g. the joint states for the torso with

```
rostopic echo /torso/joint_trajectory_controller/state
```

and add or subtract these values to the ones already defined as `torso_*_ref` in the `cob_calibration_data/cob3-X/calibration/calibration.urdf.xacro`.

4.3.2 Automatic calibration

The automatic calibration step calibrates can be used for intrinsic and extrinsic camera calibration, as well as for hand eye calibration. The manual calibration steps from section 4.3.1 are a prerequisite for the automatic calibration. For more information and tutorials about the automatic calibration, see [33].

4.4 Backup and Restore

4.4.1 Backup the entire system

We recommended to backup your system when you have a stable software version, e.g. all hardware drivers setup and running. You can backup the whole disks of your robot to an external hard disk using the tool dd.

Be sure hat the external device hat enough free space as an ext4 partition, you

[33] `http://www.ros.org/wiki/cob_calibration`

can format it using gparted [34] . With the new partition mounted in your system execute the following command:

```
sudo dd if=/dev/sdaX of=/dev/sdbY
```

where /dev/sdaX is the local partition where ubuntu is installed and /dev/sdbY is the partition where your external device is mounted. With this command you copy the whole partition, this step will take several hours depending on the disk size.

4.4.2 Restore the entire system

with the following instructions you can restore your system to a previous backed up version. However you should be aware of that if backing up and restoring fails you will need to setup your system from scratch. So we only reccomend to restore your system if nothing else helps to get the system up and running again.

If you have a backup on an external hard disk you can use a CD or USB stick with live linux to restore the system with the following command:

```
sudo dd if=/dev/sdbY of=/dev/sdaX
```

where /dev/sdbY is the partition where your external device is mounted and /dev/sdaX is the local partition where you want to restore ubuntu to.

[34] http://gparted.sourceforge.net/

Chapter 5

Support

5.1 General support

Please consult ros answers[1] to see if your problem is already known.

Please please contact your local administrator or use the mailing list[2] for additional support or feature discussion.

5.2 Report a bug

If you discover a bug in one of the Care-O-bot stacks, please fill a ticket on our trac system[3]. A view of active tickets can be found at [4].

[1] http://answers.ros.org/questions/?tags=care-o-bot,cob_driver&start_over=true

[2] http://www.care-o-bot-research.org/contributing/mailing-lists

[3] http://www.care-o-bot-research.org/trac

[4] http://www.care-o-bot-research.org/trac/report/1

5.3 Useful tools for debugging and working with the robot

In this section we introduce some useful tools which can be used while working with the robot to facilitate the source code management.

5.3.1 Modifying and developing code

If you want to modify code from existing stacks, we recommend to create an overlay of the stack in you home directory. To facilitate this process we have created a *githelper* script which automatically sets up your git environment with generating ssh-keys, uploading them to github, forking stacks (if necessary) and cloning stacks to your machine. You can either install a read-only version from our main fork (*ipa320*) or insert your own username and password to fork and clone your own version of the stack.

You can get the script with

```
wget https://raw.github.com/ipa320/setup/master/githelper
chmod +x githelper
```

The script can be used by simply typing

```
./githelper --help
```

5.3.2 Working with git

If you are not used to it, working with git is difficult in the beginning. Nevertheless it is a great tool which helps a lot managing decentralised development of our source code. `www.github.com` offers a reliable hosting service and offers tools for graphical visualisation or merging pull requests. We recommend everybody to do some basic git tutorials which can be found on various places in the web.

To make your live with git a little easier we have created a tool called `githelper`, which allows you to do git operations like getting the status, pushing, pulling and even merging in an easy way over multiple repositories at the same time. To know what githelper can do type

```
./githelper --help
```

5.4 FAQ

In this section we try to answer some frequently asked questions. If your question is not covered, but you think it is relevant for others too, please contact fmw@ipa.fhg.de.

5.4.1 Working with the robot

The robot doesn't move when I press a button on the command_gui. Make sure that the emergency stop is released properly, see section 2.4. To inspect which component is failing have a look at your dashboard, see section 2.5.3.1.

Reaction time of the robot slows down Make sure you don't hit the boarder of either CPU, IO or network load on all pcs. You can use commands from the ROS cheat sheet[5]. Especially the following commands can be useful while inspecting your system (you'll find documentation about this tool in the cheat sheet):

```
roswtf
rosnode ping
rosnode machine
rostopic bw
rostopic hz
rxgraph
```

Additionally to the ROS specifiq tools you can use standard linux tools to inspect your system:

```
top
iotop
gnome-system-monitor
```

[5] http://www.ros.org/wiki/Documentation?action=AttachFile&do=get&target=ROScheatsheet.pdf

Some tips and tricks: In general you should try to distribute CPU, IO and network load between the robot pcs in a way that during normal operation you don't come close the limits. E.g. keep the CPU load lower than 75% and overall network traffic lower than 75% of the maximum bandwidth of your ethernet router (have in mind that there are different maximum rates for cable-based and wireless ethernet). Possible ways to reach this is

- Reduce computing effort with optimizing implementation for your algorithms
- Put nodes together on the same machine which need to either
  - communicate in a high rate
    (e.g. laser scanners, base controller and navigation, check frequency with `rostopic hz`)
  - exchange huge amount of data
    (e.g. camera driver and object detection, check with `rostopic bw`)
- for data flow triggered nodes, reduce frequency of input data stream
  (e.g. don't use full frame rate of cameras as input if your detection algorithm if it is not capable of calculating each frame anyway, e.g. reduce 30Hz frame rate of kinect to 1-5Hz for object detection)
- use a mechanism to start and stop components (e.g. stop object detection if no object detection is needed at the moment to reduce CPU load and consider unsubscribing from topics to reduce network load.)

5.4.2 Developing on the robot

I have modified code or developed new code, how can others use it? Use the pull request feature at `www.github.com`, for help see [6].

I have new code, where should I put it? We already have a big variety of stacks containing different functionalities. An overview of all stacks which belong to the bringup layer are listed in section 2.2. Except for new drivers there shouldn't be a need for you to add new packages to this stacks.

[6] `http://help.github.com/send-pull-requests/`

Above the bringup layer there are several stacks for navigation, manipulation and perception functionalities. It is hard to say in which stack you can put your code, therefore please contact fmw@ipa.fhg.de to discuss where your code fits best.