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 Thursday 10th May, 2012

Contents

Nomenclature

X ..... robot number, e.g. X=3 for cob3-3

Y ..... pc number, e.g. Y=2 for pc2

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 3) addresses robot administrators and covers topics like setting up the pcs, configuring network and add new user accounts. The third part (chapter 4) 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

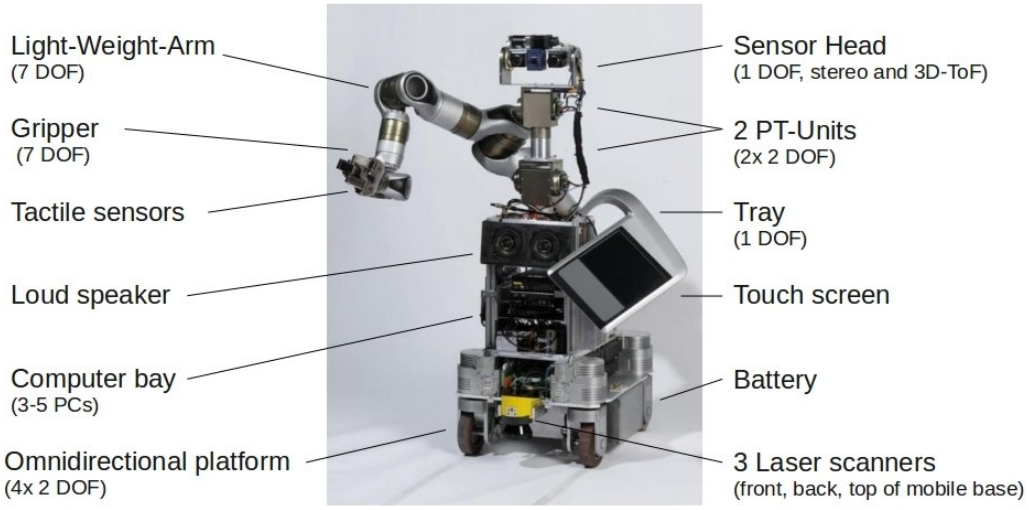
NOTE: Before you start working with the robot you have to attend a safety introduction. If you haven't got one yet, please contact your local robot administrator.

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.

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

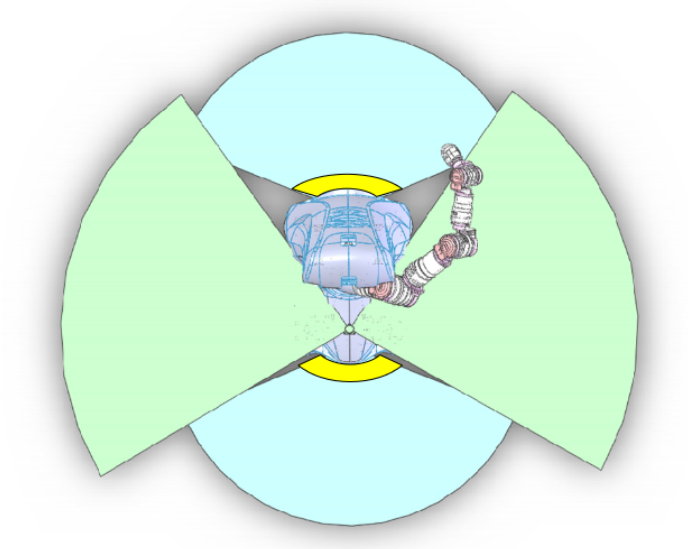
Please be aware that not all robot movements are safe for you as an user, the robot itself and the environment. Therefore please don't hesitate to activate the emergency stop in situations which are not foreseen by the user. There are three ways of stopping the robot: On the robot you have two red buttons 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.

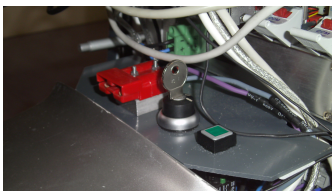
NOTE: If you hear a "click" while releasing the emergency stop, but the dashboard and command_gui still show that the emergency stop is activated, turn the key to position II again and hold it for some seconds until the dashboard or command_gui show no more emergency stop.

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 3.1.7). 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 brinup.

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
```

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

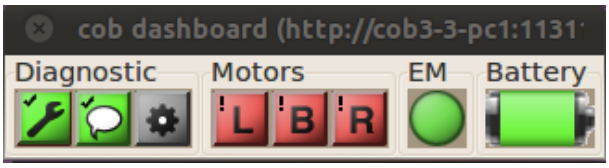
```
roslaunch cob_bringup dashboard.launch
```

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

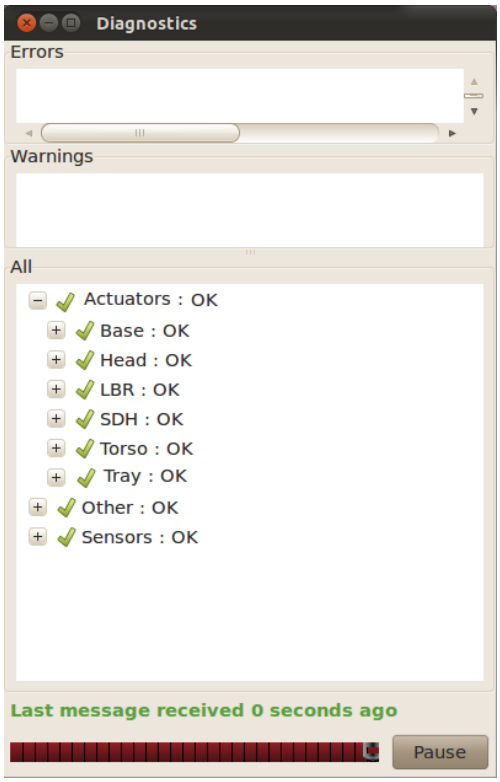
command_gui and offers a wide range of buttons to move the robot to predefined positions using low level control commands. It also offers buttons for initialising and recovering the actuators. The emergency stop status is visible in the upper left corner of the command_gui. Before initialising or moving 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:

In this screen-shot 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.

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
```

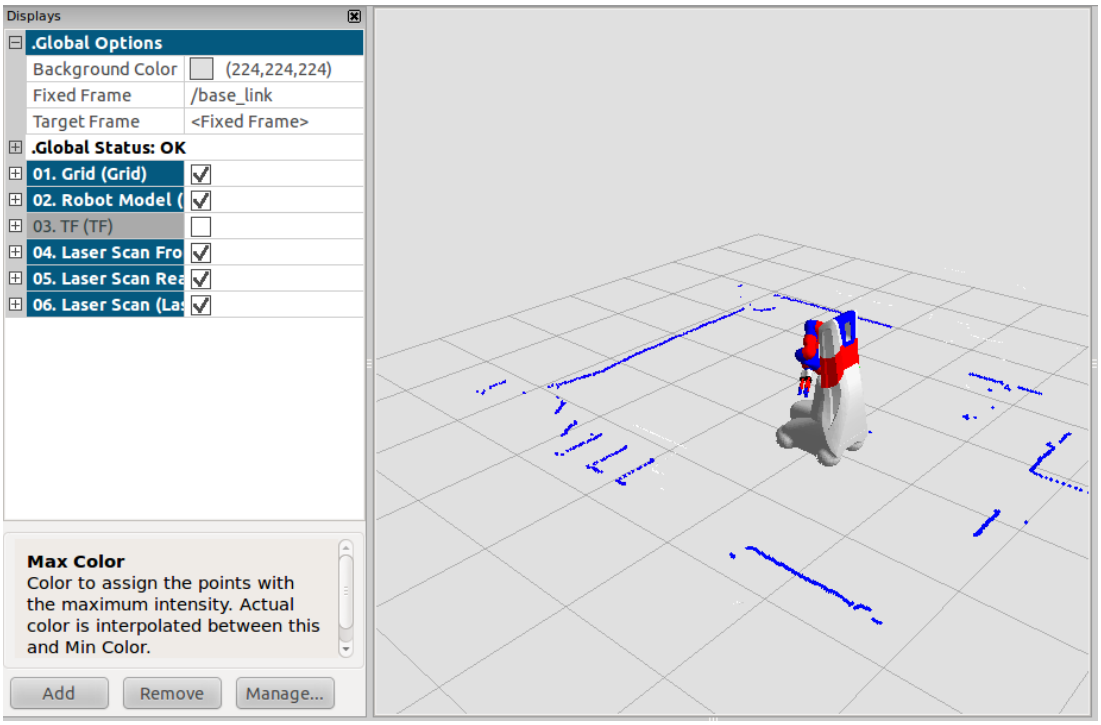
[4] `http://www.ros.org/wiki/rviz`

```
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:



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.
- For moving the torso: Hold the dead-man button and the upper or lower neck button, then use the up_down or left_right axis to move the torso.

- For moving the tray: Hold the dead-man button and the tray button, then use the up_down axis to move the tray.
- For moving the arm: Hold the dead-man button and one of the arm buttons, then use the up_down or left_right axis to move the selected arm joints.

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 turn the key to position I, disconnect the power cable and turn off the power supply.

2.7 Packing the robot

tbd

Chapter 3

Administrator manual

3.1 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 coveres 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.

3.1.1 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 10.4 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`.

3.1.2 Install basic tools

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.

```
sudo apt-get update
sudo apt-get install vim tree openssh-server gitg meld curl
```

To facilitate the further setup we created a setup repository with some helpful scripts. To checkout the setup repository use:

```
mkdir ~/git
cd ~/git
git clone git://github.com/ipa320/setup.git
```

3.1.3 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 `192.168.0.1` 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 `192.168.0.101` for pc1 and `192.168.0.102` for pc2. Enable portforwarding for port `2201` to `192.168.0.101` and for port `2202` to `192.168.0.102`.

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. 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 pc's.

3.1.3.1 Network configuration on pc1

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.0.101 # internal ip adress of pc1
netmask 255.255.255.0 # netmask

auto eth1
iface eth1 inet static
address 192.168.42.1 # ip adress for controller network
netmask 255.255.255.0 # netmask
```

3.1.3.2 Network configuration on pc2

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.0.102 # internal ip adress of pc2
netmask 255.255.255.0 # netmask

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

3.1.4 Install NFS

After the network is configured properly we can setup a NFS netween the robot pcs. pc2 will act as the NFS server and pc1 as NFS client.

3.1.4.1 NFS configuration on pc2 (server)

Install the NFS server package and create the NFS directory

```
sudo apt-get install nfs-kernel-server
sudo mkdir /u
```

Add the following following line to `/etc/fstab`:

```
/home    /u        none      bind      0         0
```

Now we can mount the drive

```
sudo mount /u
```

Activate `IDMAPD` in `/etc/default/nfs-common` by changing the `NEED_IDMAPD` to yes

```
NEED_IDMAPD=yes
```

Copy the file ∼`/git/setup/nfs_setup/server/exports` to `/etc/exports`

```
cp ~/git/setup/nfs_setup/server/exports /etc/exports
```

Change the home directory of the `robot` user from `/home/username` to `/u/username` in the `/etc/passwd` file.

After finishing you need to reboot the pc

```
sudo reboot
```

3.1.4.2 NFS configuration on pc1 (client)

Install the NFS client package and create the NFS directory

```
sudo apt-get install nfs-kernel-server autofs
sudo mkdir /u
```

Activate `IDMAPD` in `/etc/default/nfs-common` by changing the `NEED_IDMAPD` to yes

```
NEED_IDMAPD=yes
```

Edit `/etc/auto.master` and add

```
/-       /etc/auto.direct
```

Copy the file ∼`/git/setup/nfs_setup/client/auto.direct` to */etc/auto.direct*

```
cp ~/git/setup/nfs_setup/client/auto.direct /etc/auto.direct
```

Activate the NFS

```
sudo update-rc.d autofs defaults
sudo service autofs restart
sudo modprobe nfs
```

Change the home directory of the `robot` user from `/home/username` to `/u/username` in the `/etc/passwd` file.

After finishing you need to reboot the pc

```
sudo reboot
```

3.1.5 Setup NTP time synchronitation

Install the ntp package

```
sudo apt-get install ntp
```

3.1.5.1 NTP configuration on pc1 (NTP server)

Edit `/etc/ntp.conf`, change the server to `0.pool.ntp.org` and add the restrict line

```
server 0.pool.ntp.org
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap
```

3.1.5.2 NTP configuration on pc2 (NTP client)

Edit `/etc/ntp.conf`, change the server to `cob3-X-pc1`:

```
server cob3-X-pc1
```

3.1.6 Install ROS and driver software

For general instructions see [1].

[1] `http://www.ros.org/wiki/Robots/Care-O-bot/electric`

3.1.6.1 Install ROS and additional tools

```
sudo apt-get install openjdk-6-jdk zsh terminator
sudo apt-get install python-pip
sudo pip install -U rosrelease
sudo apt-get install ros-electric-care-o-bot-robot
```

3.1.6.2 Creating an overlay for stacks

If the release version of the stacks are not working for you, you can install overlays for individual stacks on the `robot` user account. To facilitate this process we have created a `create_overlay.sh` script which automatically generates ssh-keys, uploads them to github, forkes the stack (if necessary) and clones it 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. The script can be used by simply typing

```
cd ~/git/setup
./create_overlay.sh [stack]
```

All stacks needed for the bringup layer are listed at section 2.2.

Note: It should typically only be necessary to create an overlay for two stacks, the `cob_robots` and the `cob_environments` stack. All other stacks should be used from their release version. If you need to modify a stack to work for you, please send us a pull request to `ipa320` on `www.github.com`.

3.1.6.3 Setup bash environment

We setup a special bash environment to be used on the Care-O-bot pcs. The environments differ from pc1, pc2 and pc3. Copy the `cob.bash.bashrc.pcX` to `/etc/cob.bash.bashrc` on each pc.

```
sudo cp ~/git/setup/cob-pcs/cob.bash.bashrc.pcX /etc/cob.bash.bashrc
```

All users have a pre-configured bash environment too, therefore copy `user.bashrc` to `~/.bashrc`

```
cp ~/git/setup/cob-pcs/user.bashrc ~/.bashrc
```

At the bottom of your `.bashrc` you have to define `ROS_MASTER_URI` to be `http://cob3-X-pc1:11311`, `ROBOT` to be `cob3-X` and `ROBOT_ENV` to point to your environment.

If you logout and login again or source your ∼`/.bashrc`, you should see different terminal colors for each pc and the `ROS_PACKAGE_PATH` should be configured. If you check

```
roscd cob_bringup
pwd
```

you should end up in `/u/robot/git/care-o-bot/cob_robots/cob_bringup`.

3.1.6.4 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 udev rules In order to have fixed device names we setup udev rules for Care-O-bot. Copy the udev rules from the setup repository to `/etc/udev/rules.d` on pc1:

```
sudo cp ~/git/setup/udev_rules/01-cob.rules /etc/udev/rules.d
```

To activate these changes you have to restart the system.

Build the bringup level packages Before setup the components you can build all the bringup level packages:

```
rosmake --rosdep-install cob_bringup
```

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[2] `libpcan` and ESD[3] `libntcan` can be used. Installation instructions can be found in the package documentation of libpcan

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

http://www.ros.org/wiki/libpcan and libntcan http://www.ros.org/wiki/libntcan.

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 [4].

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 laser_front.launch
```

To check if there is some data published use

```
rostopic hz /scan_front
```

Check the rear scanner in the same way.

tbd, setup safety region: use windows based sick toolbox

Hokuyo URG laser scanner tbd

Relayboard tbd

Base You have to configure the elmo controllers, all the information about the driver is in [5], you find the parameters of this configuration in cob_hardware_config package in config/base.

Tray sensors The tray sensors are defined in udev_rules and the library libphidgets[6] for this component is already built in the bringup level.

[4] http://www.ros.org/wiki/cob_sick_s300
[5] http://www.ros.org/wiki/cob_base_drive_chain
[6] http://www.ros.org/wiki/libphidgets

Schunk SDH with tactile sensors There is only one specific Firmware and library version supported. The version number is listed on [7]. Add the robot user to the group `dialout` in the `/etc/group` file.

Schunk powercubes (lwa, torso, tray) 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[8]. A sample configuration can be fount at [9].

The ROS configuration is done in `cob_hardware_config`, e.g. in `/config/lwa.yaml` or `/config/torso.yaml`. Documentation about the ROS parameters can be found at [10] and [11].

Head axis Configuration in `cob_hardware_config` in `/config/head.yaml`. Documentation about parameters can be found at [12].

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 [13].

You have to setup the network interface on the robot pc2 to access the cameras, please refer to section 3.1.3.

Kinect The kinect is connected to pc2.

---

[7] `http://ros.org/wiki/schunk_sdh`

[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.ros.org/wiki/cob_head_axis`

[13] `http://ros.org/wiki/cob_camera_sensors/AVT_Prosilica`

3.1.7 Create a new user account

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
```

After that you can add a new user. On pc2 and with administration rights you can use the following instruction

```
cd ~/git/setup
sudo ./cobadduser new_user_name
```

All you need to do now is to login as the new user, create an ssh-key and add the key to the authorized keys to be able to login to all pcs without password. This is necessary for launching nodes remotely on all pcs. The following lines are an example for cob3-3:

```
ssh-keygen
ssh-copy-id cob3-X-pc1
ssh cob3-X-pc1
ssh cob3-X-pc2
```

3.2 Network infrastructure for external access to the robot

For the robot internal network setup please refer to section 3.1.3.

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 192.168.0.101
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.

3.2.1 Setting up your building network (recommended)

Setting up you building network to enable dns and port forwarding to the internal network.

3.2.2 Manual setup for each remote pc

You can setup a route to the internal network addresses. Please change the robot name and your network device to fit your settings. E.g. for connecting to `cob3-X` on `eth0`

```
sudo route add -net 192.168.0.0 netmask 255.255.0.0 gw cob3-X dev
    eth0
```

for connecting to `cob3-X` on `wlan0`

```
sudo route add -net 192.168.0.0 netmask 255.255.0.0 gw cob3-X dev
    wlan0
```

For name resolution you will probably have to edit the file `/etc/hosts` on the robot pcs as well as on the remote pc. Therefore add the following addresses to the `/etc/hosts` of your remote pc.

```
192.168.0.101 cob3-X-pc1
192.168.0.102 cob3-X-pc2
192.168.0.103 cob3-X-pc3
```

Add your ip adress and hostname to the `/etc/hosts` of all robot pcs.

3.3 Calibration

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

3.3.1 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_controller/state
```

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

3.3.2 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 3.3.1 are a prerequisit for the automatic calibration. For more information and tutorials about the automatic calibration, see [14].

3.4 Backup and restoring users

tdb

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

Chapter 4

Support

4.1 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.

4.2 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

4.3 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.

4.3.1 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 *create_overlay.sh* script which automatically generates ssh-keys, uploads them to github, forkes the stack (if necessary) and clones it 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. The script can be used by simply typing

```
cd ~/git/setup
./create_overlay.sh [stack]
```

4.3.2 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 -h
```

4.4 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.

4.4.1 Working with the robot

The robot doen'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.

4.4.2 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 [5].

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.
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.

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