KUKA youBot User Manual

version 1.02

February 6, 2013

Contents

Chapter 1

Introduction

The KUKA youBot is a mobile manipulator that was primarily developed for education and research. We hope that it will become a standard platform for the community that is interested in education and research in mobile manipulation.

The KUKA youBot comes with fully open interfaces and allows the developers to access the system on nearly all levels of hardware control. It further comes with an application programming interface (KUKA youBot API), with interfaces and wrappers for recent robotic frameworks such as ROS [1] or OROCOS [2], with an open source simulation in Gazebo and with some example code that demonstrates how to program the KUKA youBot. The platform and the available software shall enable the user to rapidly develop his/her own mobile manipulation applications.

Giving full access to nearly all levels of control of the robot comes at a price, however. It allows the customer to access and control each of its actuators and sensors directly as he or she thinks is best. Customers can develop applications whose functionalities are only limited by the KUKA youBot's kinematic structure, its drive electronics, and motors. Such applications could damage the robot itself or other objects in its workspace or could result in harm of a human being or an animal.

The KUKA youBot qualifies as a so-called **partly completed machinery** [3]. You will notice that the KUKA youBot does not come with any preinstalled software. All software will have to be installed by the customer him-/her-self (from a Live-USB stick delivered with the robot) to turn the KUKA youBot into an operational machine. This is to emphasize that it is solely the responsibility of the user to turn to KUKA youBot into a safe machine. We refer to the chapter on safety for more information.

A great deal of the software will be open source software available on KUKA youBot store. In fact, whatever software we describe in this document is open source and we will try hard to promote the development of open source software for mobile manipulation in general and, of course, especially for and around the KUKA youBot. This is the front side of the coin. The back side is that nobody will take any liability for the proper functioning of such software. Neither will KUKA nor Locomotec.

Neither KUKA nor Locomotec will take any liability for damages or harms caused by a KUKA youBot operated by any third party software. By operating third party products (software and hardware) or own hardware and software the customer takes the full responsibility and liability for damages and harms caused by a KUKA youBot. This is the inevitable price for providing full access to nearly all levels of the robot hardware and control system, which is a request of the robotics research community that lead to the development of the KUKA youBot.

Although we often call the KUKA youBot a desktop mobile manipulator, we do not recommend operating it on a desktop, unless it is secured by some border around it. Without safety precautions it may fall off the table and smash your cat or dog, or even worse, your feet or toes. If you operate the KUKA youBot on a table or desktop the manipulator may also hit your upper body or head and severely injure yourself. Please, be aware of that. If you still do it, you do it on your own risk.

Furthermore, we do not recommend putting the robot into a microwave or any other oven, into a bathtub or under a shower. We do not recommend putting it under water or exposing it to any other robot-unfriendly or hostile environment. Such actions would destroy your youBot.

For further safety instructions and instructions on the proper use of the hardware we refer to the respective section in this document and to the KUKA youBot hardware documentation [4].

The purpose of this manual is to explain the usage of existing software and support developing new

software. It will be revised, updated and possibly expanded in regular intervals. On KUKA youBot store you will always find the latest release of this document as well as the latest releases of software around the KUKA youBot. We do not take any liability for outdated information. Please, always refer to the latest releases of documentation and software.

To work with and develop software for the KUKA youBot requires some basic knowledge in robot manipulation, mobile robotics, and mobile manipulation. It further requires some basic understanding of the operating system Linux (Ubuntu). You need to know how to open a console window, start programs and alike. For help and support please refer to Chapter 10 "Service and support".

1.1 Short system overview

The KUKA youBot consists of two main parts:

- The KUKA youBot **omni-directional mobile platform** consists of the robot chassis, four mecanum wheels, motors, power and an onboard PC board. Users can either run programs on this board, or control it from a remote computer. The platform comes with a Live-USB stick with preinstalled Ubuntu Linux and drivers for the hardware. Details of the software are described below.
- The KUKA youBot **arm** has five degrees of freedoms (DOF) and a two-finger gripper. If connected to the mobile platform, the arm can be controlled by the onboard PC. Alternatively, the arm can be controlled without the mobile platform by using an own PC connected via Ethernet cable.

Additional sensors can be mounted on the robot. This manual will provide you with information on the sensors working with the KUKA youBot, so far.

1.2 Quickstart

The following steps will enable you to control the robot with a joypad.

1. Read the safety instructions as described in Chapter 9.
2. Switch off the robot in case you had already powered it on.
3. Plug the bootable Live-USB stick to one of the empty USB ports on the long side of the robot.
4. Connect a mouse, a keyboard and a monitor to the USB and VGA ports on the long side of the robot.
5. Place the robot in some free space with a minimal radius of one meter so that the robot will not collide with any object once it starts moving.
6. Switch on the robot (cf. Section 2.1).
7. A boot screen should appear. Choose the option "Live CD" and confirm by pressing `return`.
8. The robot's operating system will boot from the USB stick.
9. A login screen will appear. Click on the name "youbot" and enter the password `youBot`.
10. Next, the Ubuntu desktop should come up.
11. Move the arm roughly into home position as described in Chapter 3.
12. Open a new terminal window by clicking the menu item `Applications > Accessories > Terminal`. The menu is located at the top of the desktop.
13. In the new terminal window enter `cd /home/youbot/applications/bin` to navigate to the applications' folder.
14. Start the demo application with `sudo ./YouBot_HelloWorldDemo`

15. The youBot will move approx. 10 cm forwards, then approx. 10 cm backwards, then approx. 10 cm to the left and approx. 10 cm to the right. After that it will unfold the arm and fold it again.

NOTE:

If you receive a "No socket connection on eth0" the Ethernet interface might have another number e.g. eth1. You can check the available interfaces by typing the command `ifconfig` in a terminal. In order to correctly specify the interface number, open the configuration file

`/home/youbot/youbot_driver/config/youbot-ethercat.cfg`

and edit the line `EthernetDevice = eth0` accordingly. Further details on the configuration files can found in Section 5.2.2.

Chapter 2

KUKA youBot hardware overview

This chapter gives a brief overview of the hardware components of the KUKA youBot. A more detailed hardware documentation can be found in the KUKA youBot hardware manual [4].

2.1 KUKA youBot omni-directional mobile platform

The KUKA youBot base is an omni-directional mobile platform with four mecanum wheels. Figure 2.1 illustrates the attached base (coordinate) frame, as it will be used in the KUKA youBot API (cf. Section 5.2). It is located in the center of odometry. Positive values for rotation about the z axis result in a counterclockwise movement, as indicated by the blue arrow. The wheel numbering for the mecanum wheels is also shown.

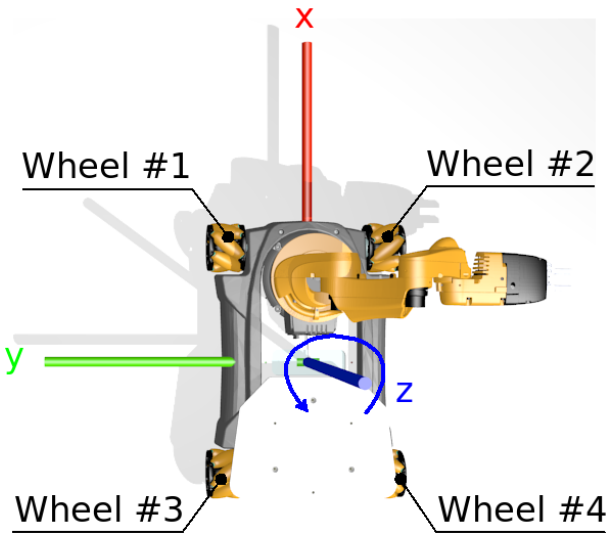

Figure 2.1: Overview of the KUKA youBot base. The Figure illustrates the attached base frame. Positive values for rotation about the z axis result in a counterclockwise movement, as indicated by the blue arrow.

The motor controllers of the four omni-directional wheels can be accessed via Ethernet and EtherCAT. The Ethernet cable can be either plugged into the onboard PC or to an external computer. The plug is a standard network cable plug that fits into each standard Ethernet port.

On the top side of the KUKA youBot base you will find a panel with power plugs (labeled as 24V OUT) and Ethernet ports (labeled as EtherCAT 1 and EtherCAT 2) for up to two KUKA youBot arms. The third power plug (labeled as 24V IN) can be used to power the KUKA youBot and recharge the battery (at the same time). There is a single Ethernet slot (labeled as Ethernet) for connecting the onboard PC to a LAN via network cable. A LCD screen will indicate the battery status. If you want to connect the internal EtherCAT controllers to an external PC you have to open the cover of the base on

the right side, unplug the Ethernet cable and replug it to an extension cable that goes into the external computer.

With the power switch you can either start the power board only – which makes sense if you want to use an external computer – or start both, the power board and the onboard PC:

- To start the power board only, push the On/Off button until the display flashes up.
- To start the onboard PC (after having started the power board), push the On/Off button again until the display flashes up and shows `PC on`.

To shutdown the KUKA youBot's onboard PC push the On/Off button until the display flashes up and indicates `PC off`. To completely shutdown the robot keep pushing the button.

2.2 KUKA youBot arm

The KUKA youBot arm is a serial chain with five revolute axes. The end effector is a two-finger gripper, that can be removed. Figure 2.2 illustrates the basic kinematic structure. Similar as for the base the motor drivers for the individual joints can be accessed via Ethernet and EtherCAT.

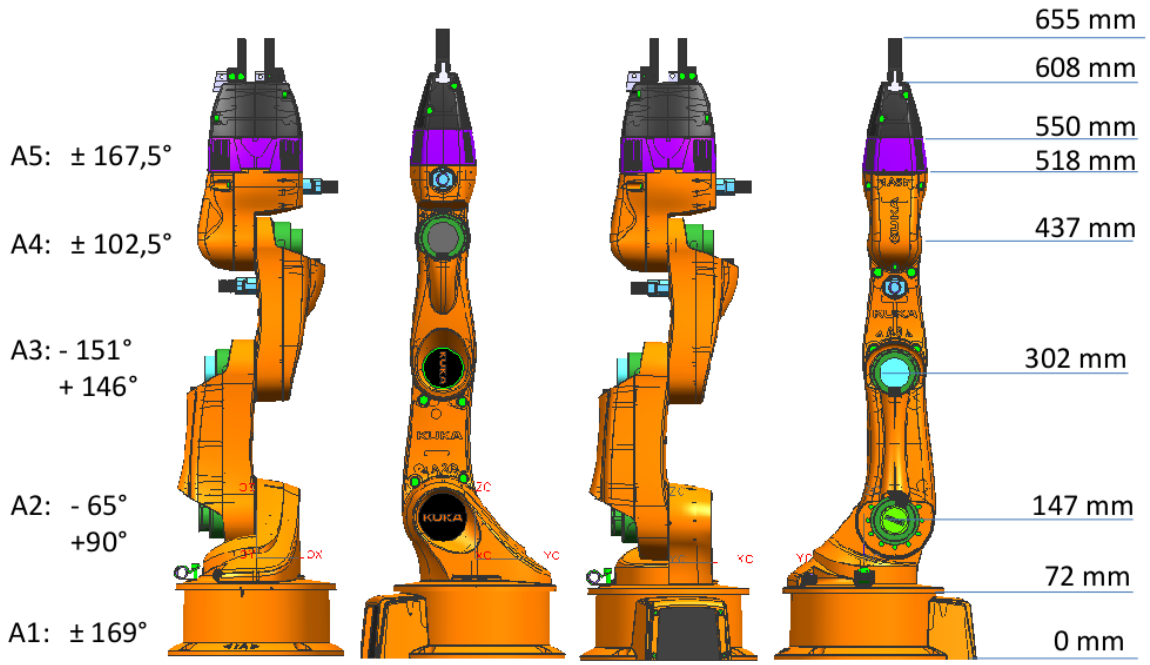

Figure 2.2: Overview of the kinematic structure of the arm. The Figure illustrates joints with limits, and the length of links between joints.

NOTE:
Even if you have a complete platform with base and arm(s), there is only one EtherCAT plug that you need to insert to either onboard or external PC.

2.3 Onboard PC

Each KUKA youBot has a Mini-ITX board as onboard computer. The main characteristics are summarized in Table 2.3. Further details can be found in Advantechs AIMB 212 datasheet [5].

Processory	Intel AtomTM Dual Core D510 (1M Cache, 2 x 1.66 GHz)
Memory	2GB single-channel DDR2 667MHz
Graphics	Embedded Gen3.5+ GFX Core, 400-MHz render clock frequency, up to 224 MB shared memory
Harddrive	32GB SSD drive

Table 2.1: Main characteristics of the onboard PC

2.4 Connecting I/O devices

It is possible to work locally on the KUKA youBot PC by connecting a keyboard, a mouse and a monitor to its ports. This is the recommended and easiest way for the initial installation. You may further have to resort to this configuration when you face problems with the network connectivity.

Mouse and keyboard must have USB connectors. For the monitor a VGA cable is required. Please, note that the onboard PC has neither PS2, DVI nor HDMI connections. If you want to use such interfaces you may have to use appropriate adapters.

Chapter 3

Moving the arm to its home position

The joints of the KUKA youBot arm are equipped with position encoders to measure joint angles. The used encoders are relative position encoders, which means, they do not yield an absolute joint angle but a angular displacement relative to a reference position. We also call these positions *home positions* or *rest positions* to indicate the arm should start its movement from these positions. As natural reference positions we use the mechanical stops of the joint limits. Once the arm approaches these joint limits it has to be moved **very slowly and gently** – no matter whether it is moved manually or by a program – to avoid any damages of the arm. In the home position the arm looks like it is folded together.

WARNING:
Only if the arm starts from the home position, it will function properly, that means its motion will be as planned (by a program). If you forget to move the arm into the home position before it starts, you may accidentally move or push one or several joints against their mechanical stops, in the worst case even at high velocities. There is a very high risk that the arm gets damaged by such an operation and you lose your warranty. The motor controllers will recognize when this happens and limit the current. But the damage may already have happened. **Never forget to put the arm into the home position, each time before you start a program using the arm!**

The homing procedure can be executed as long as the joint motors are not powered (i.e. no current is applied). That is always the case when the robot is turned off. It should also be the case after a program has finished its operation properly. However, programs have the tendency to crash every now and then. It then may happen that the joint motors remain under power and cannot be moved. In such cases, it is recommended to turn off and reboot the whole robot in order to avoid damage.

To bring the arm into the home position follow the procedure described below and graphically shown in Figure 3.1. It is recommended but not requested that the robot is shut off, while you perform this procedure. Under no circumstances the am must be under power.

1. First, **gently** move joints #2, #3, and #4 manually to their mechanical stops as illustrated in Figures 3.1(a) to 3.1(c). Once you have done that the arm looks like it is folded.
2. Next, **gently** move joint #1 counterclockwise (seen from top) until you reach its mechanical stop (see Figures 3.1(d)).
3. Finally, **gently** move joint #5 counterclockwise until you reach its mechanical stop (see Figures 3.1(e)).

Once all joints are in their home positions (at their mechanical stops) your program or application can safely use these positions as reference for their further motion and start its operation. Once your program is finished the actual joint positions are lost. They will not be stored in some registers, since they can be changed manually without any program noticing it.

Figure 3.1: Procedure to move the manipulator into the home position: all joints are moved to their mechanical stops. Figures (a) to (c) show how to move the joints 2, 3 and 4 to the home position. Figures (d) to (f) illustrate how to rotate the first joint. Figures (g) to (i) show how to rotate the gripper.

Chapter 4

Live-USB Stick

The Live-USB stick that comes with the KUKA youBot is a bootable device that contains an Ubuntu distribution, basic drivers and software for the KUKA youBot to let you run the robot. This section explains in detail how to use the stick for testing the robot, how to install its content to the onboard PC, and how to create a new stick if required.

The stick runs on most x86 PCs and is not restricted to the KUKA youBot onboard PC. This is especially relevant for customers who have purchased a KUKA youBot arm as a stand alone system.

NOTE: It may happen that the Live-USB stick does not work on the very latest PC hardware. The reason for that may be the missing support within the used kernel. In this case the user is requested to manually install the required software as described in chapter 5.

4.1 Booting from Live-USB stick

It is possible to work directly on the operating system booted from the Live-USB stick. This can be a convenient method to quickly test the robot hardware. However, in general it is not recommended to use the USB stick as working environment; in that case you are advised to install the stick on the onboard PC, or an external PC, as described in the next section.

In order to boot from the Live-USB stick for testing the robot, follow these steps:

1. Configure the BIOS of your PC so that you can boot from USB stick. If you want to boot from the onboard PC instead you can skip this step.
2. Connect the USB stick.
3. When starting to boot from USB stick, you will get a boot prompt. Just press `Enter`.
4. Next you will get a menu, in which you can select how to boot the Ubuntu operating system. Every choice, except `Installation`, will start the Ubuntu live version directly from the USB stick.
5. After the start, you will see a login screen. There is a panel at the bottom of this screen, where you can set some configurations like language, keyboard etc. Select the settings according to your needs.
6. You can now logon to the system as user “youbot”. The password is also “youbot”. This user belongs to the `sudoers` group, so (s)he has administrative rights in combination with the `sudo` command.

Booting from a USB stick may take a while. During the boot process, the resolution of the VGA-output will change a few times. If your monitor does not support this resolution, you will just see a black screen. This is not a failure and the booting process is still going on. After a few minutes the resolution should change back. So do not restart your system too early, if this happens.

4.2 Installing from Live-USB stick

You can permanently install the software provided on the Live-USB stick onto the onboard PC of the KUKA youBot or on an external PC. To do that you may use the default installation procedure [6] provided with the stick.

NOTE: Before you start the installation on a PC of your own, you should create a backup of the data on your hard disk. You should also be familiar with creating partitions before making any changes.

You will not erase any data if this is your first installation on the KUKA youBot onboard PC, because the KUKA youBot is shipped with an empty harddrive. The boot procedure is similar to the one described in the previous section:

1. Configure the BIOS of your PC, so that you can boot from USB stick.
2. Connect the USB stick.
3. When starting the boot procedure from the USB stick, you will get a boot prompt. Just press `Enter`.
4. Select the `Installation` item in the menu to install the operating system to the hard-drive.
5. Apart from one aspect, related to the user account, the installation routine that will be started is the standard Ubuntu installation routine. This exception is described below. During the installation routine a number of windows will pop up and require certain decisions to be made and certain information to be entered. The windows and the solicited information is described in the following:
   (a) *Language*: Select your desired language and click `Install Ubuntu 10.04 LTS`.
   (b) *Where are you?*: Select the location that is closest to your position and click `Forward`.
   (c) *Keyboard layout*: Select the correct keyboard layout. Click `Forward`.
   (d) *Prepare disk space*: If you want Ubuntu to use your entire hard drive then select `erase`, select the hard drive that you want to install Ubuntu and click `Forward`. Note that installing Ubuntu on your entire hard disk will erase all data on the drive! If you want to install Ubuntu on a single partition Dual Booting, Select `Guided>resize`. In the `New partition size` area, drag the area between the two partitions to create your desired partition sizes. Click `Forward`.
   (e) *Who are you?*: This is the exception compared to the standard Ubuntu installation routine. You have to set some arbitrary name and password for the user, but this user will not be added to the users list. The Live-USB stick will always add the user "youbot" with password "youbot" automatically to the system. If you want to create other users, you have to add them manually after the installation process. Nonetheless you have to complete this step. It is impossible to skip the *Who are you?* window. The only one thing you can change here is the name of the PC. The default name is `youbot-pc`.
   (f) Finally, verify that the language, layout, location, and personal information are correct and click `Install`. The installation wizard will begin.
   (g) Once the installation wizard is finished, the *Installation complete* window will appear. Click `Restart now` to restart your computer. Ubuntu is now installed. Before your system reboots, you should unplug the Live-USB stick.

4.3 Setting up a new Live-USB stick

If you want to set up a new USB stick with the latest version of the remastered Ubuntu ISO image, please follow these instructions:

1. Download the ISO image from the KUKA youBot website.
2. Plug an empty USB stick with sufficient capacity (> 4GB) in your PC. **Note that all previous stored data will be lost!**
3. Select a tool to create a bootable USB stick. Find below two suitable tools that are able to create bootable USB sticks.

4.3.1 Using usb-creator

One method to create a bootable USB stick is to use the `usb-creator` tool [7] (Linux only). The Ubuntu USB startup disk creator is available in the packages `usb-creator` and `usb-creator-gtk` (`usb-creator` and `usb-creator-kde` on Kubuntu).

You can find it in `System>Administration>Create a USB startup disk` (Ubuntu Desktop) or `K-Menu>Applications>System>Startup` Disk Creator (Kubuntu). If you do not find it there then you can install it using the Synaptic Package Manager, or by entering the following command in the terminal:

```
$ sudo apt-get install usb-creator usb-creator-gtk
```

Then start `usb-creator` and follow the steps below:

1. In its top panel, click `Other` and locate and select the .iso file that you have downloaded.
2. Insert the USB flash drive into your computer. You do not need to mount it or view it. It should show up in the bottom pane titled `Disk to use`. (You may have to use GParted to format the USB drive; ext3 will definitively work)
3. Make sure you have the correct device selected before clicking the button `Make Startup Disk`. Your USB-flash-drive should now be ready for use.

4.3.2 Using UNetbootin

Another method for creating bootable USB sticks is the UNetbootin [8] software available for Windows and Linux. You can download the binary/executable from `http://unetbootin.sourceforge.net/`. If you are using Windows:

1. Run the executable file.
2. Select an ISO file by selecting option `Disk Image`.
3. Select an USB drive as target drive and type.
4. Then press `Ok` and the creation will start. This might take a while.
5. You do not need to restart your PC. Just click `Finish`.

If you are using Linux:

1. Make the binary executable (using either the command `chmod +x ./unetbootin-linux-<YOUR_VERSION>`, or going to `Properties>Permissions` and check `Execute`).
2. Then start the application. You will be prompted for your password.
3. Select an ISO file by selecting option `Disk Image` and install target (USB drive or hard disk).
4. Select an USB drive as target drive and type.
5. Than press `Ok` and the creation will start. This might take a while.
6. You do not need to restart your PC. Just click `Finish`.

Chapter 5

Installing from scratch

The next sections explain the installation procedures for various hardware drivers and the KUKA youBot API (*application programming interface*). Although the KUKA youBot API and all currently available hardware drivers are pre-installed on the Live-USB stick we explain how to install (download, resolve dependencies, compile and build) this software from scratch. This will also enable the user to modify the configuration and to add other hardware drivers.

5.1 Operating system

The recommended operating system for the KUKA youBot is Ubuntu Linux. All drivers and applications provided by Locomotec are tested on this distribution. Currently, Ubuntu version 10.04 (Lucid Lynx) is supported. Please check the KUKA youBot Store for support of newer versions.

Although at present Linux is the only operating system that runs on KUKA youBot, this is not a binding decision for the future. As a matter of fact, support of a number of other operating systems such as Microsoft Windows is planned. Android can already be used by linking the Android device to the youBot's PC running Linux.

5.2 KUKA youBot API

The KUKA youBot driver stack consists of two parts:

1. **EtherCAT driver**: On the lowest (accessible) level of control all actuators of the KUKA youBot are driven over the EtherCAT protocol [9]. This protocol uses standard network hardware like Ethernet adapters, respective RJ45 connectors and network cables to connect EtherCAT master and slaves and establish the communication between them (see EtherCAT specs for details [9]). The KUKA youBot utilizes an open source driver implementation SOEM [10] for the master.

2. **KUKA youBot API**: On top of the EtherCAT driver stack there is a second software layer – the KUKA youBot API – to command the youBot platform. This layer was developed by the Autonomous Systems Group at Bonn-Rhein-Sieg University of Applied Sciences, Germany. It builds upon SOEM EtherCAT drivers and provides additional functionality relevant to platform kinematics and joint level commanding. Currently, this library provides the basic functionalities to set the Cartesian velocities of the base, to read odometry data from motor encoders etc. It furthermore provides the functionality to command the manipulator on joint level (setting and getting joint configurations/data, gripper). This part will be expanded very soon by libraries such as OROCOS KDL [2] to also command the youBot arm in Cartesian space.

5.2.1 Installing the youBot API

There are two different ways of installing the KUKA youBot API. While downloading the source files is always the same, it can be compiled either using `rosmake` or `CMake`. Both methods are described in the following. In this process the required EtherCAT drivers are installed as well, providing the complete driver stack. There are no pre-compiled packages available so far.

Downloading the source files

First you need to get the source files for the KUKA youBot API which include the EtherCAT drivers as well:

```
$ cd /home/youbot
$ git clone https://github.com/youbot/youbot_driver.git
```

This will download the sources in the default folder `/home/youbot/youbot_driver`.

Compiling with `rosmake`

Compiling with `rosmake` is usually a bit more convenient compared to calling `CMake` directly, thus it is the recommended way.

This method requires an installed instance of ROS on your system. For further information about how to install `ROS` itself see chapter 8.1.1 or the ROS webpage[1].

Note that only the build system of ROS is used here. If you don't want to use any other features of `ROS` later it is sufficient to install only the `ros-base` packages. It is important to add the `/home/youbot/youbot_driver` folder to the ROS_PACKAGE_PATH environment variable as described in chapter 8.1.1.
After that you can compile the driver:

```
$ rosmake youbot_driver
```

After the build process has finished without failures the driver is ready for use. If the command `rosmake` is not found, `ROS` was not installed properly. If the package `youbot_driver` can not be found, it was probably not added correctly in the ROS_PACKAGE_PATH environment variable.

Compiling using `CMake`

You can also call `CMake` directly to compile the youbot drivers. This might be needed in cases when ROS is not available on the machine. First download the driver sources as described above. In order to compile enter in a command shell:

```
$ cd youbot_driver
$ mkdir build
$ cd build
$ cmake ..
$ make
```

If no error occurs the driver is now ready to use. In order to let other applications find it, you need to add an environment variable `YOUBOTDIR` that points to the driver's folder:

```
$ gedit ~/.bashrc
```

Insert the following line:

```
export YOUBOTDIR=/home/youbot/youbot_driver
```

To make the change effective just start a new command window or enter

```
$ source ~/.bashrc
```

[1] `http://www.ros.org/wiki/electric/Installation/Ubuntu`

Installing for multiple users

After compiling the driver as described above, the KUKA youBot API is ready to be used. It resides in the folder `/home/youbot/youbot_driver` and can be found by other application either via the environment variables `ROS_PACKAGE_PATH` (if `ROS` is used) or `YOUBOTDIR` (otherwise). This directory contains all the software that is required for application development, configuration and execution, including headers (`youbot_driver/include`), libraries (`youbot_driver/lib`), configuration files (`youbot_driver/config`) and manuals.

If other users than "youbot" want to create and start youBot programs as well, they can just refer to the installation in the youbot's home folder by setting the environment variables in the same way. The advantage is that all users share the same configuration files for the robot.

If a user wants to use his own configuration files, he is also free to download and compile the KUKA youBot API in his own home folder. There are no issues with this solution. The user only has to be aware to eventually update and configure the new instance of the driver as needed.

In addition the `CMake` files for the KUKA youBot API allow to install the driver in system folders. This requires root priviliges for the user.

NOTE: Installing the driver as in the following is not recommended when using `ROS`. Otherwise there will be two instances of the driver possibly found, one in the system folder and one by the `ROS_PACKAGE_PATH` variable. This may cause confusion and strange errors if any changes are made to one of them.

If you still want to install in a system folder issue the commands

```
$ cd /home/youbot/youbot_driver/build
$ sudo make install
```

In that case it is not necessary to set the `YOUBOTDIR` variable, as the `CMake` files looking for the driver will find it directly.

5.2.2 Configuration files

Before starting to use the KUKA youBot for the first time, you need to edit some configuration files first:

- `/home/youbot/youbot_driver/config/youbot-ethercat.cfg`
- `/home/youbot/youbot_driver/config/youbot-base.cfg`
- `/home/youbot/youbot_driver/config/youbot-manipulator.cfg`
- `/home/youbot/youbot_driver/config/youbot-manipulator2.cfg`

The `youbot-ethercat.cfg` contains basic information about the EtherCAT connection, and will always be used when calling the KUKA youBot API. The configuration files for the base and manipulator are only accessed if the respective classes in the API for arm and manipulator are used.

The youbot-ethercat.cfg configuration file

The most important parameter in this file is called `EthernetDevice`, and in most cases the only one you have to take care of. It defines the ethernet device name of the PC at which the platform or the manipulator is connected to:

```
EthernetDevice = eth0
```

To find out the name of the device, open a terminal and run the following command:

```
$ ifconfig
```

This will list the network configuration including the names of all devices. Usually the Ethernet device is something like eth0, eth1, etc..

The youbot-base.cfg and youbot-manipulator.cfg configuration files

The `youbot-base.cfg` file must be edited if an omni-directional platform is being used. Analogously, the `youbot-manipulator.cfg` file is important when using a KUKA youBot arm. In each file you only have to edit the `[JointTopology]` section, where you can set the index for each joint. This index internally relates to the slave devices in the EtherCAT network, whereas each motor controller has its own a slave device. The `[JointTopology]` establishes a mapping between the EtherCAT slave devices and the topological joint numbers. The enumeration of the joints is quite simple. The EtherCAT master starts counting the first slave device for the joints with number 1, the second one with number 2 and so on. The following examples depict correct configurations:

If you only have a mobile platform, than set the following values in `youbot-base.cfg` (**only platform**):

```
[JointTopology]
BaseLeftFront  = 1
BaseRightFront = 2
BaseLeftBack   = 3
BaseRightBack  = 4
```

If you only have a KUKA youBot arm, set the following values in `youbot-manipulator.cfg` (**only arm**):

```
[JointTopology]
ManipulatorJoint1 = 1
ManipulatorJoint2 = 2
ManipulatorJoint3 = 3
ManipulatorJoint4 = 4
ManipulatorJoint5 = 5
```

If you have both, a platform together with an arm, you have to edit both files. In case your PC is directly connected to the platform, and your platform is connected to the arm (default setting), set the following values:

```
#in youbot-base.cfg
[JointTopology]
BaseLeftFront  = 1
BaseRightFront = 2
BaseLeftBack   = 3
BaseRightBack  = 4

#in youbot-manipulator.cfg
[JointTopology]
ManipulatorJoint1 = 5
ManipulatorJoint2 = 6
ManipulatorJoint3 = 7
ManipulatorJoint4 = 8
ManipulatorJoint5 = 9
```

If you are using a KUKA youBot platform with a dual arm system, you have to add one more configuration file for the second arm. The `JointTopology` section in that file will be incremented accordingly to reflect the EtherCAT enumerations. A valid setup for all files is the follwing:

```
#in youbot-base.cfg
[JointTopology]
BaseLeftFront  = 1
BaseRightFront = 2
BaseLeftBack   = 3
BaseRightBack  = 4
```

```
#in youbot-manipulator.cfg
[JointTopology]
ManipulatorJoint1 = 5
ManipulatorJoint2 = 6
ManipulatorJoint3 = 7
ManipulatorJoint4 = 8
ManipulatorJoint5 = 9

#in youbot-manipulator2.cfg
[JointTopology]
ManipulatorJoint1 = 10
ManipulatorJoint2 = 11
ManipulatorJoint3 = 12
ManipulatorJoint4 = 13
ManipulatorJoint5 = 14
```

Please note, assigning physical EtherCAT slave devices in the configuration files will determine which arm in your application is the "first" or the "second" arm. You are free to select this by interchanging the values of the files `youbot-manipulator.cfg` and `youbot-manipulator2.cfg`.

5.2.3 Installing the youBot applications

After the installation and configuration of the driver is finished, you can now install the applications to run your KUKA youBot the first time. Here we differentiate as well between compiling with `rosmake` or only `cmake`.

Although you can use `rosmake` to install the sources, all of the applications mentioned here do not require `ROS` at all. That means these programs are standalone C++ programs that are linked directly with the youbot driver. If you are also interested to develop your own standalone applications, we recommend to use the `hello_world_demo` app as template. In this case `rosmake` is only a tool to make the installation easier. It is only an option, not a must. You can also proceed the installation of the standalone applications with using `cmake` only.

In both cases you have to download the applications first:

```
$ cd /home/youbot
$ git clone https://github.com/youbot/youbot_applications.git applications
```

Compiling with `rosmake`

In order to compile with `rosmake` the folder of the application must be included in the `ROS_PACKAGE_PATH` variable, as described in section 8.1.1. `rosmake` will compile all applications with the command:

```
$ rosmake applications
```

After that go into one specific application folder. There you will find a bin directory. Execute the binary as root:

```
$ cd /home/youbot/applications/<folder_of_application>/bin
$ sudo ./<name_of_binary>
```

Compiling with `cmake`

You can also use `cmake` to compile the standalone applications. The main difference is that you have to configure and build all applications separately. The following shows an example for the `hello_world_demo` application:

```
$ cd /home/youbot/applications/hello_world_demo
$ mkdir build
$ cd build
$ cmake ..
$ make
```

To run the applications go to the bin folder and execute the binary as root:

```
$ cd /home/youbot/applications/<folder_of_application>/bin
$ sudo ./<name_of_binary>
```

5.3 Generic Sensor Drivers

The KUKA youBot is designed as an open system. It should be therefore rather easy to extend the system and add new hardware components, for example, new sensors. The only requirements that the user has to face are the compatibility of such components with the available interfaces and buses (Ethernet/EtherCAT, USB, ...) and the availability of drivers (low-level as well as higher levels). For instance, if the user wishes to extend the robot with a USB or FireWire camera then it is required that usbcore, uvcvideo, ieee1394 etc., drivers are installed.

In the following we describe the sensors and drivers which are available for the KUKA youBot at present.

- Vision sensors - USB cameras, FireWire cameras
- Rangefinders - Hokuyo
- 3D vision sensors - Kinect

5.3.1 Generic 2D Cameras

Todays Linux distributions come with precompiled kernel modules for common USB and IEEE1394 FireWire cameras. Therefore, in most cases it is not necessary to install them separately. In cases where the kernel does not include a driver for the camera that should be used with the KUKA youBot, first the standard needs to be identified that the respective camera complies with.

- **USB cameras**: For USB cameras there exist several of such standards such as UVC, V4L, VISCA and alike. They all have their own software stack and procedures how to install. Please check for the respective documentation. The high-level API `unicap` supports many monocular cameras (UVC, V4L, IEEE1394). `unicap` is included in `robotpkg` MPD and can be installed with `robotpkg`
- **FireWire cameras**: There are two versions of driver software stacks that support FireWire under Linux. The older version is based on IEEE1394, OHCI1394, DC1394 etc driver and library stack. The code name of the newer version is Juju. You have to make sure compatibility between kernel and installed driver version. This is also critical since the `udev` device manager creates different device nodes for old and new stacks. In the old stack device nodes often will have the form of `/dev/video1394/*`, while the new stack will have the form `/dev/fw*`. One possible way to access FireWire devices is through the `unicap` library or the the `coriander` application.

In addition to standard standalone library type of access to devices and device programming, one can also use the same devices through pre-installed robot software frameworks. These frameworks also use the same or similar low/high level drivers. The advantage is that these frameworks often come with extra functionalities for inter-process communication, AI, robotics, vision etc. The next chapter provides more detailed information on these frameworks for the KUKA youBot.

5.3.2 Bumblebee2 Stereo Camera

The *Bumblebee2* manufactured by *Point Grey Research* is an IEEE1394 (FireWire) stereo vision camera. It has a base line of $0.12m$. Further details about the camera can be found in [11].

Installation

The Bumblebee2 camera works with standard FireWire drivers, as described above. We use the libdc1394 driver stack:

```
$ sudo apt-get install libavc1394-dev libraw1394-dev libraw1394-doc
$ sudo apt-get install libdc1394-22-dev libdc1394-22-doc
```

Usage

Before starting the camera you have to load the kernel modules as follows:

```
$ sudo modprobe dv1394
$ sudo modprobe video1394
$ sudo modprobe raw1394
```

Set permissions for the camera:

```
$ sudo chmod 666 /dev/raw1394
```

Depending on your FireWire controller, this must be also done for `/dev/video1394/0` or `/dev/video1394/1`. You can use `coriander` to display images. To install this application you can type:

```
$ sudo apt-get install coriander
```

First plug the camera, then start coriander. To display the images:

```
$ coriander
```

1. On the `Camera` tab click on `Restart global ISO Control`.
2. Go to tab `Services` and click on `Restart ISO control` button in the `ISO control section`.
3. Then choose `Format7, Mode_3` in the `Format section`.
4. Click on `Display` button to start video capture.
5. Go up and down with `BPP` in the `Options` section. 8bit means left image and 16bit means right image.

5.3.3 Hokuyo Laser Range Finder

The *URG-04LX-UG01* is a small laser range finder with an approximate maximal scanning distance of $5.6m$. It has a scan angle of 240 degree and can acquire scans with a frequency of 10.0Hz.

Installation

The Hokuyo laser scanner essentially works by sending and receiving commands over the SCIP protocol. The *URG-04LX-UG01* has an USB connection, and is treated as a serial device. That means the Linux kernel already provides drivers for such devices. It will be added per default as `/dev/ttyACMx` whereas x stands for a running number. Further documentation can be found here:
`http://www.hokuyo-aut.jp/02sensor/07scanner/download/urg_programs_en/index.html`
To check if the Hokuyo is found you can type `dmseg`. You should have something similar like this:

```
$ dmesg
... (abbreviation) ...
[ 2822.168000] usb 3-1: new full speed USB device using uhci_hcd and address 2
[ 2822.328000] usb 3-1: configuration #1 chosen from 1 choice
[ 2822.440000] cdc_acm 3-1:1.0: ttyACM0: USB ACM device
[ 2822.444000] usbcore: registered new interface driver cdc_acm
[ 2822.444000] /build/buildd/linux-source-2.6.22-2.6.22/drivers/usb/class/cdc-acm.c:
 v0.25:USB Abstract Control Model driver for USB modems and ISDN adapters
```

Usage

The manufacturer of Hokuyo offers a sample application (among others) to check, if the Hokuyo sensor works. It is called `vmon`. To use it download vmon_linux.zip
`http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/vmon_linux.zip` and follow the install instructions. They can be summarized as follows:
Unpack the archive and go to the `vmon_linux` folder:

```
$ sudo apt-get install libboost libboost-date_time
$ sudo apt-get install libboost-filesystem libboost-thread libxmu6 libxxf86vm1
$ sudo dpkg -i libscipcomm_0.2_i386.deb vmon_0.2_i386.deb
```

Start the application by typing:

```
$ sudo vmon
```

Finally right click and select *Connect* and then "/dev/ACM0" (assuming the device has number 0). Figure 5.1 illustrates the vmon application.

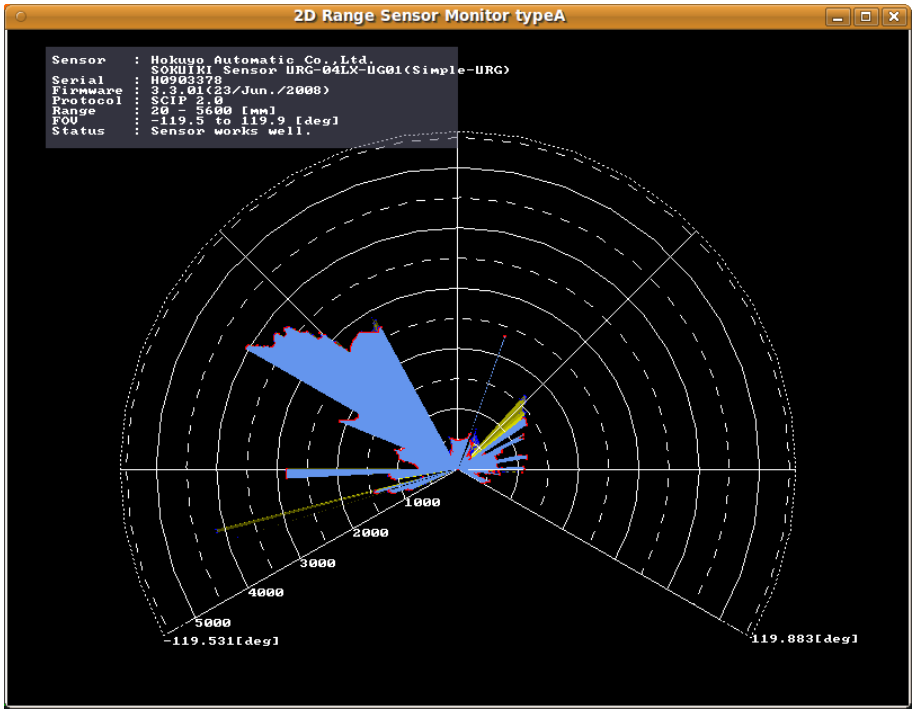

Figure 5.1: Illustration of the vmon application for the Hokuyo laser range finder.

By default an user application will need root permissions, so you can use `sudo` (as seen above) or manually set permissions:

```
$ sudo chmod a+rw /dev/ttyACM0
```

Instead of setting permissions (each time) of `ttyACM*` devices a *udev* rule can be used. To do so create a new group `usb` and add the user `<username>` to this group. Of course `<username>` needs to be replaced by the actual user name of the logged in user, e.g. *youbot* in case of the default system configuration for the onboard PC of the KUKA youBot.

```
$ sudo groupadd usb
$ sudo usermod -aG usb <username>
```

Then create a new *udev* rule for Hokuyo devices:

```
$ sudo touch /etc/udev/rules.d/47-hokuyo.rules
```

Add the following line, save and replug:

```
SUBSYSTEMS=="usb", KERNEL=="ttyACM[0-9]*", ATTRS{manufacturer}=="Hokuyo Data Flex for USB",
ATTRS{product}=="URG-Series USB Driver", MODE="0666", SYMLINK+="sensors/hokuyo"
```

This `udev` rule has two effects: a) it always grants correct permissions and b) it creates a constant system link: `/dev/sensors/hokuyo` which can be uses instead of `/dev/ttyACM*`. The latter one would change each time the Hokuyo is replugged.

NOTE: This behavior might not be intended in case of two or more Hokuyos. A good starting point to add Hokuyo system links with unique IDs can be found here:
`http://www.ros.org/wiki/hokuyo_node#Using_udev_to_Give_Hokuyos_Consistent_Device_Names`

5.3.4 SwissRanger 3D Camera

The *Mesa Imaging SwissRanger SR4000 3D Camera* is a Time-of-Flight 3D sensor. It utilizes a phase shift principle of emitted and received infrared light to measure depth to the surfaces in the viewing frustum. The sensor has a range between approx. $0.8m$ and $5m$. Further details can be found in [12].

Installation

Download and install the latest `libmesasr-dev` driver. You can find the drivers here:
`http://www.mesa-imaging.ch/drivers.php`.
They are offered as debian packages e.g. `libmesasr-dev-1.0.14-665.i386.deb`.

Next type

```
dpkg -i libmesasr-dev-1.0.14-665.i386.deb
```

The version number might change.

Usage

Check if the camera works with `libMesaSRTester`:

```
$ sudo /usr/bin/libMesaSRTester
```

1. Select `Basic test`
2. Select `scan`
3. Now the device should be listed. Press `2` to start the test.

Optionally you can grant the driver the required permissions to start an application with user rights. To do so create a new group `usb` and add the user `<username>` to this group. Of course `<username>` needs to be replaced by the actual user name of the logged in user, e.g. `youbot` in case of the default system configuration for the onboard PC of the KUKA youBot.

```
$ sudo groupadd usb
$ sudo usermod -aG usb <username>
```

Then create a new `udev` rule for SwissRanger devices:

```
$ sudo touch /etc/udev/rules.d/46-mesa.rules
```

Add the following lines to the `/etc/udev/rules.d/46-mesa.rules` file

```
BUS!="usb", SUBSYSTEM!=="usb_device", ACTION!="add", GOTO="kcontrol_rules_end"

#Swissranger SR3k(old),SR3k,SR4k
SYSFS{idVendor}=="0852", SYSFS{idProduct}=="0074", MODE="666", GROUP="usb"
SYSFS{idVendor}=="1ad2", SYSFS{idProduct}=="0074", MODE="666", GROUP="usb"
SYSFS{idVendor}=="1ad2", SYSFS{idProduct}=="0075", MODE="666", GROUP="usb"

LABEL="kcontrol_rules_end"
```

Finally replug the device.

5.3.5 Kinect 3D Camera

The *Kinect* from *Microsoft* is a 3D camera, that uses an infrared pattern projector and an infrared camera to infer distance from that captured pattern. It comes with a color web cam, a microphone array, an accelerometer and a tilt unit. It has a detection range of approx. $1.2m$ to $3.5m$.

Microsoft does not offer any drivers for Kinect, but there are two sources for getting a driver. The first one is an industry consortium *OpenNI*[2] consisting of PrimeSense among others, who manufacture the underlying hardware of the Kinect. This consortium offers an open source middleware for "Natural Interaction" devices, including support for the Kinect.

The other source is the `libfreenect`. It is a reverse engineered driver by the `OpenKinect`[3] project. `libfreenect` is a rather lightweight library to get access to depth and color images. The following install instruction refers to this driver.

Installation

First satisfy dependencies, if not yet installed:

```
$ sudo apt-get install libglut3-dev libxmu-dev libxi-dev libusb-1.0-0-dev
```

Navigate to a folder of your choice, checkout the latest version from the git repository and compile it with `cmake` and `make`:

```
$ git clone https://github.com/OpenKinect/libfreenect.git
$ cd libfreenect
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
```

Usage

The `libfreenect` comes with the OpenGL based sample application `glview`. Make sure the Kinect is properly connected, e.i. the USB is plugged and the $12V$ power supply is established. Go to the folder where the driver has been compiled and start it with root permissions. Figure 5.2 shows a screen shot the `glview` application.

```
$ cd <path_to_driver>/libfreenect/build/bin/
$ sudo ./glview
```

[2] http://www.openni.org/
[3] http://openkinect.org/wiki/Main_Page

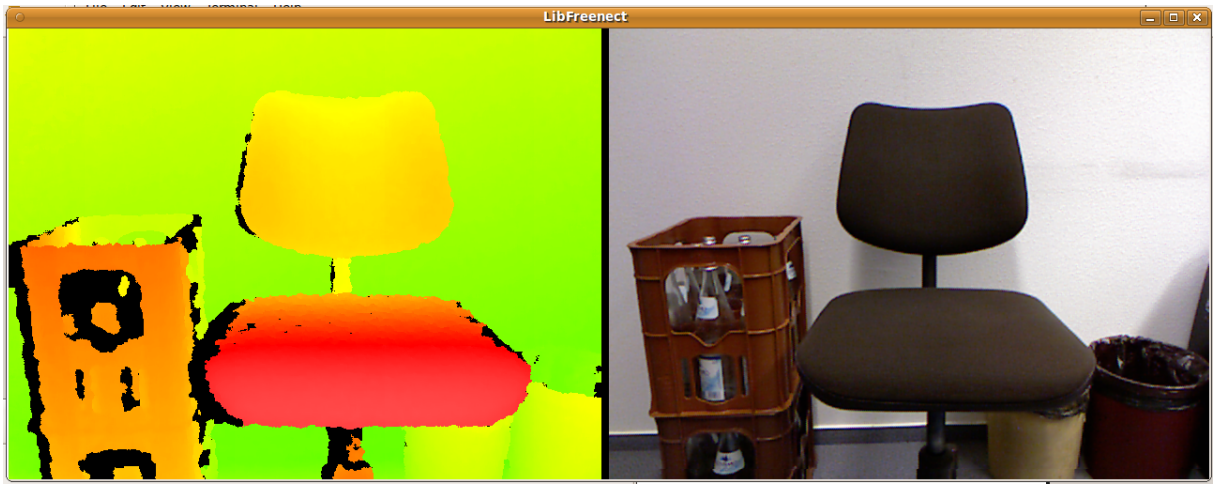

Figure 5.2: Illustration of the glview application that comes with the libreenect driver.

Chapter 6

Working with the KUKA youBot

In this chapter you find information on how to work with the KUKA youBot, once everything is installed and configured. It is assumed that the system is setup as provided on the Live-USB stick. If you made any changes to the default installation, be aware that you might have to adjust the following instructions appropriately to match your case.

6.1 Local login with keyboard and monitor

The login procedure with a keyboard and monitor connected is as follows:

1. Make sure the onboard PC is turned off.
2. Plug the mouse/keyboard and the monitor to the PC board connections on the long side of the robot.
3. Switch on the robot (see Section 2.1).
4. The Linux system should boot up and after a short while show you a login screen.
5. Click with the mouse on the *youbot* picture in the center.
6. A blinking cursor for entering the password should appear. Enter the password `youbot`.
7. If the password was entered correctly, you should see the Ubuntu desktop.

If you connect the I/O devices without switching off the robot before, the Linux system may not be able to identify the connected monitor correctly which in turn may lead to a blank screen.

In general it is not required to logout manually before turning off the robot. You should, however, make sure that you save all files and documents that you have worked on or else your data and changes might be lost.

6.2 Account settings

On the default KUKA youBot Linux installation, one user account is setup:

Account: youbot
Password: youbot

Use the same password `youbot` for getting administrative rights, for example for changing system files or using the `sudo` command. This password is also needed to start the KUKA youBot drivers.

If multiple people will be working with the robot, you may consider creating individual new user accounts. All drivers and programs will work with other accounts as well, but you have to take care of the different home directories that each account will use.

6.3 Folder Structure

The default folder structure for the KUKA youBot is organized as follows:

```
/home/youbot/                          # the youBots's home folder
/home/youbot/youbot_driver             # folder to the API
/home/youbot/youbot_driver/config      # folder for the config files
/home/youbot/applications              # folder for applications using the youBot API
/home/youbot/ros_stacks                # folder for the ROS wrapper of the KUKA youbot API
/opt/ros/<ros_version>                 # default ROS installation folder
```

You can use any new folder, as you like, for storing your own projects and applications.

6.4 Updating the system

In this section you will find information on how to update the software for the KUKA youBot. All updates have to be triggered manually as there are no automatic checks running on the system whether updates are available.

6.4.1 Operating system

To update the packages of the operating system we recommend to use `apt-get`. The update procedure can be initiated with one command:

```
$ sudo apt-get update
```

6.4.2 KUKA youBot API

To update the KUKA youBot API, i.e. the youbot driver, just follow these instructions:

```
$ cd /home/youbot/youbot_driver # change to directory, where the youbot_driver is installed
$ git pull                      # get the newest sources

# now recompile the new sources
$ rosmake                       # recompile using rosmake
                                # or
$ cd build                      # recompile using cmake only
$ rm -rf *
$ cmake ..
$ make
```

On the default KUKA youBot system both methods work identically.

6.4.3 youBot applications

Updating the applications is similar to the update of the youbot driver. Follow these instructions:

```
$ cd /home/youbot/applications  # change to directory, where the applications are installed
$ git pull                      # get the newest sources

# now reinstall the new sources
$ rosmake                       # reinstall using rosmake
                                # or
$ cd /<name_of_app>/build       # reinstall using cmake only
$ rm -rf *
$ cmake ..
$ make
```

In the latter case, i.e. not using rosmake, the commands have to be issued for each application separately.

6.4.4 Updating from old robotpkg based installation

Formerly (until youBot Ubuntu ISO image version 0.2.0, i.e. rougly until August 2012) another installation tool called `robotpkg` was used to install the youbot driver. Now this tool is not supported any more and we recommend to update your system if you still have `robotpkg` installed. Your system is outdated if you still have the following folder structure:

```
/home/youbot/pkgmanager             # package management tool robotpkg
/home/youbot/pkgmanager/robotpkg    # meta package database of robotpkg
/home/youbot/pkgmanager/openrobots  # installed libs, bins, etc.
```

Deleting `robotpkg`

If robotpkg is still installed, we recommend to uninstall it because it will not be supported any longer. Additionally there could be a conflict between the configuration files, libaries etc., if they are existing twice in different locations. Since `robotpkg` is only installed in the home directory, you can remove it easily by deleting the `pkgmanager` folder.
Note: If you have made any changes in the `robotpkg` or `openrobots` sources and want to keep them, you should copy these files before you remove `robotpkg`. This might in particular include the configuration files for the KUKA youBot in `/home/youbot/pkgmanager/openrobots/config`.
In order to delete robotpkg enter

```
$ cd /home/youbot/
$ rm -rf pkgmanager
```

Now you can also remove the `ROBOTPKG_BASE` environment variable from the `.bashrc`.

```
$ cd /home/youbot/
$ gedit .bashrc

# remove following entry from the .bashrc file
export ROBOTPKG_BASE=/home/youbot/pkgmanager/openrobots
```

Of course do not forget to install the new youbot driver as described in section 5.2.1.

Chapter 7

Programming with the KUKA youBot API

This chapter includes several examples which provide an introduction to programming the KUKA youBot with the KUKA youBot API.

7.1 Introduction to API

The KUKA youBot API is the programming interface through which the developer can access and control the KUKA youBot hardware. In this document we often use the short term KUKA youBot API in place of the somewhat longer term KUKA youBot OODL API. The "extension" OODL means *Object Oriented Devices Layer* and emphasizes the object oriented design of the software behind the API. This software, which is open source, may be viewed as a high-level driver for the robot. It consists of a set of decoupled functional sub-systems.

In the KUKA youBot API the robot arm is represented as 5DOF kinematic chain, and the omni-directional mobile platform is treated as a collection of revolute joints, where each joint consists of a motor and a gearbox. The KUKA youBot API uses the following three main classes:

- The class `YouBotManipulator` represents the KUKA youBot arm as an aggregation of a series of joints and a gripper (see Figure 7.1).
- The class `YouBotBase` represents the KUKA youBot omni-directional platform (see Figures 7.1, 7.2).
- The class `YouBotJoint` represents a joint either of the arm or the platform (see Figure 7.3).

The access to properties of each sub-system is organized hierarchically. For each entity (system, sub-system) represented in the API there is a set of configuration classes. These configuration classes enable one to get or set particular configurations for that entity. This is performed through `getConfiguration()` and `setConfiguration()` methods. For instance for joints this might look like as depicted in the figure 7.3.

The KUKA youBot API makes a distinction between different kinds of data provided by a joint. These are *setPoint* and *sensed data*. Each physical quantity is represented by its own class for setPoint or sensed data, which also takes into account its physical units (see Figure 7.4). The same principle applies to joint parameters, where each physical quantity (e.g. current, motor coil resistance) is represented by its own class. The number and character of parameters supported by the API is defined by the firmware which runs on a motor-controller. For more detailed information on different parameters refer to [13].

Every KUKA youBot joint is represented as a `youbot::YouBotJoint` class in the the API. At this stage the API does not make a distinction if it is a base joint which powers a wheel or a manipulator joint. `youbot::YouBotBase` and `youbot::YouBotManipulator` are aggregated classes that allow access to a particular joint instance through `youbot::YouBotJoint`.

To set and get setPoint values or read some sensors from joints, you have to use the `youbot::JointData` group of classes, e.g. `youbot::JointVelocitySetpoint` or `youbot::JointSensedCurrent`. To configure

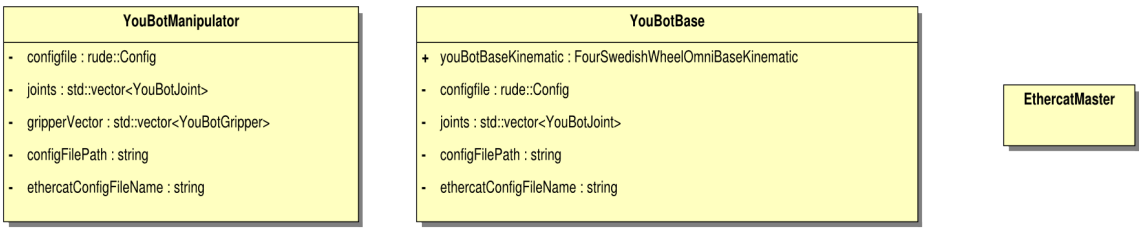

Figure 7.1: Main elements of the KUKA youBot API: YouBotManipulator and YouBotBase.

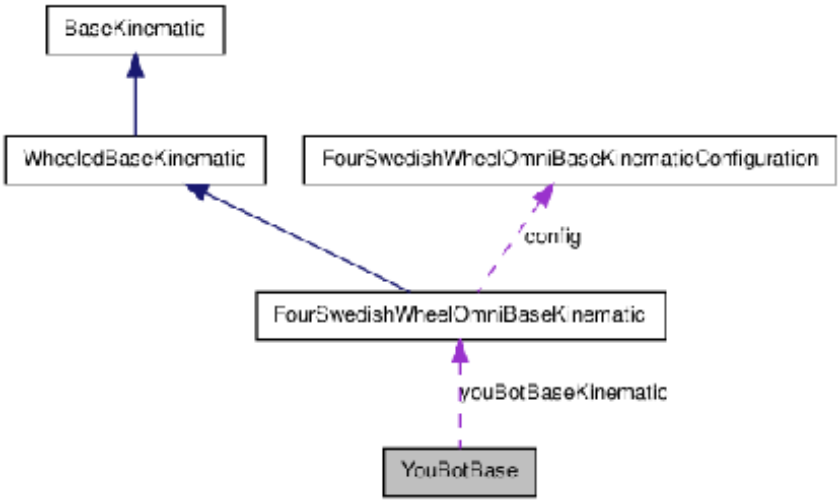

Figure 7.2: Relation graph of KUKA youBot base to base kinematics hierarchy.

parameters of a joint you have to use the JointParameter group of classes. This could be for instance `youbot::MaximumPositioningSpeed`. For more detailed information on class and methods please refer to API documentation [13]. In Chapter 7 we introduce step-by-step examples based on the KUKA youBot API.

7.2 Examples

7.2.1 Hello World

The following example can be found under `/applications/hello_world_demo`. It allows one to move the KUKA youBot mobile base as well as its arm of the KUKA youBot. In the following we are going to explain the source code step by step.

Lines 7-11 defines default values for velocities for joints. Lines 12-27 instantiate the base platform and manipulator arm with appropriate configuration values from `.cfg` files. Then three types of velocities are instantiated for the base platform. These are defined by the form of base kinematics and represent relative motion in forward and backward (longitudinal motion), sideways (transversal motion) and rotation about the center of geometry of the base (rotational motion). Based on this model, the KUKA youBot base is driven through a set of commands in lines 32-73. The same approach is applied to the KUKA youBot manipulator arm when moving its joints. The joints are position driven (lines 74-108) by the given value of the rotation angle. A first set of statements (lines 74-85) unfold the arm. The arm is folded back through the set of statements in lines 89-100.

```
1 #include "youbot/YouBotBase.hpp"
2 #include "youbot/YouBotManipulator.hpp"
3
```

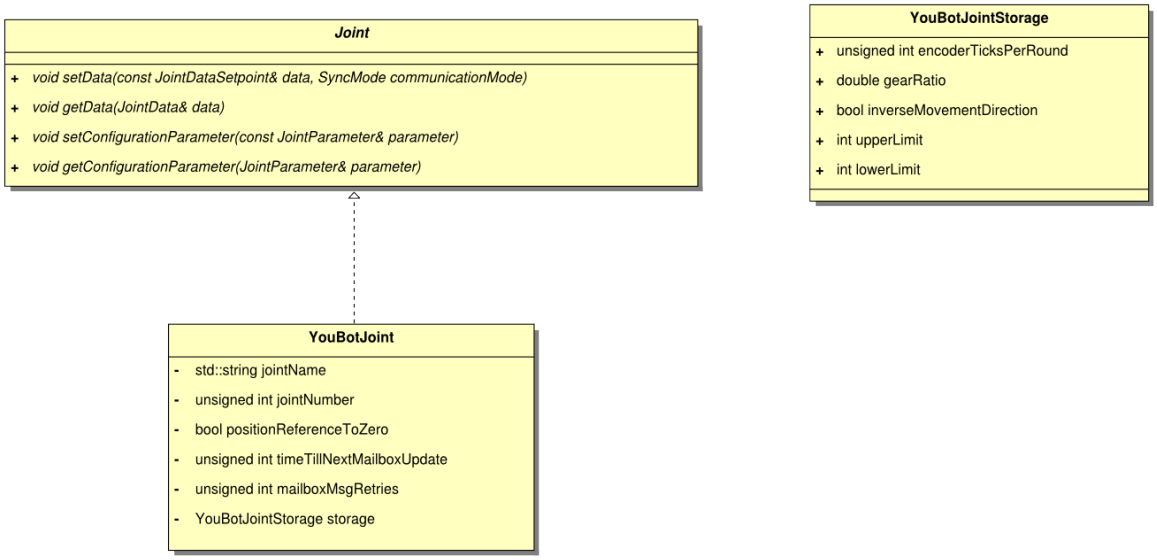

Figure 7.3: youBot joint interface.

```
4  using namespace youbot;
5
6  int main() {
7
8  /* define velocities */
9      double translationalVelocity = 0.05; //meter_per_second
10     double rotationalVelocity = 0.2; //radian_per_second
11
12     // Per default it uses the config file ../config/youbot-base.cfg
13     /* create handles for youBot base and manipulator */
14     YouBotBase myYouBotBase("youbot-base");
15
16     // Per default it uses the config file ../config/youbot-manipulator.cfg
17     YouBotManipulator myYouBotManipulator("youbot-manipulator");
18
19     /*
20      * Variable for the base.
21      * Here "boost units" is used to set values in OODL,
22        that means you have to set a value and a unit.
23      */
24     quantity<si::velocity> longitudinalVelocity = 0 * meter_per_second;
25     quantity<si::velocity> transversalVelocity = 0 * meter_per_second;
26     quantity<si::angular_velocity> angularVelocity = 0 * radian_per_second;
27
28     /* Variable for the arm. */
29     JointAngleSetpoint desiredJointAngle;
30
31     try {
32
33      /*
34       * Simple sequence of commands to the youBot:
35       */
```

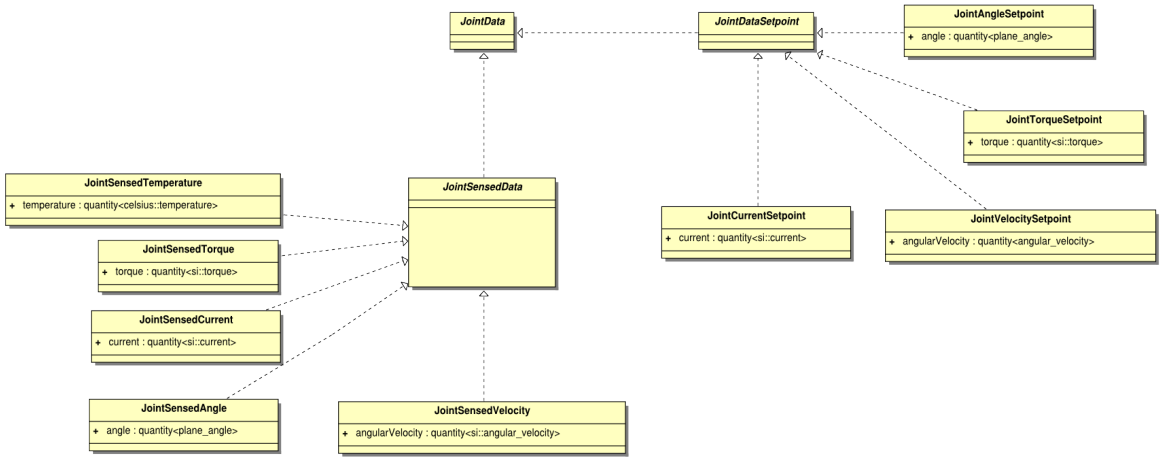

Figure 7.4: KUKA youBot joint data types.

```
36
37      /* forward */
38      longitudinalVelocity = translationalVelocity * meter_per_second;
39      transversalVelocity = 0 * meter_per_second;
40      myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
41      LOG(info) << "drive forward";
42      SLEEP_MILLISEC(2000);
43
44      /* backwards */
45      longitudinalVelocity = -translationalVelocity * meter_per_second;
46      transversalVelocity = 0 * meter_per_second;
47      myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
48      LOG(info) << "drive backwards";
49      SLEEP_MILLISEC(2000);
50
51      /* left */
52      longitudinalVelocity = 0 * meter_per_second;
53      transversalVelocity = translationalVelocity * meter_per_second;
54      angularVelocity = 0 * radian_per_second;
55      myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
56      LOG(info) << "drive left";
57      SLEEP_MILLISEC(2000);
58
59      /* right */
60      longitudinalVelocity = 0 * meter_per_second;
61      transversalVelocity = -translationalVelocity * meter_per_second;
62      angularVelocity = 0 * radian_per_second;
63      myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
64      LOG(info) << "drive right";
65      SLEEP_MILLISEC(2000);
66
67      /* stop base */
68      longitudinalVelocity = 0 * meter_per_second;
69      transversalVelocity = 0 * meter_per_second;
70      angularVelocity = 0 * radian_per_second;
71      myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
72      LOG(info) << "stop base";
73
```

```
74      /* unfold arm */
75      desiredJointAngle.angle = 2.56244 * radian;
76      myYouBotManipulator.getArmJoint(1).setData(desiredJointAngle);
77
78      desiredJointAngle.angle = 1.04883 * radian;
79      myYouBotManipulator.getArmJoint(2).setData(desiredJointAngle);
80
81      desiredJointAngle.angle = -2.43523 * radian;
82      myYouBotManipulator.getArmJoint(3).setData(desiredJointAngle);
83
84      desiredJointAngle.angle = 1.73184 * radian;
85      myYouBotManipulator.getArmJoint(4).setData(desiredJointAngle);
86      LOG(info) << "unfold arm";
87      SLEEP_MILLISEC(4000);
88
89      /* fold arm (approx. home position) */
90      desiredJointAngle.angle = 0.1 * radian;
91      myYouBotManipulator.getArmJoint(1).setData(desiredJointAngle);
92
93      desiredJointAngle.angle = 0.01 * radian;
94      myYouBotManipulator.getArmJoint(2).setData(desiredJointAngle);
95
96      desiredJointAngle.angle = -0.1 * radian;
97      myYouBotManipulator.getArmJoint(3).setData(desiredJointAngle);
98
99      desiredJointAngle.angle = 0.1 * radian;
100     myYouBotManipulator.getArmJoint(4).setData(desiredJointAngle);
101     LOG(info) << "fold arm";
102     SLEEP_MILLISEC(4000);
103
104    } catch (std::exception& e) {
105     std::cout << e.what() << std::endl;
106     std::cout << "unhandled exception" << std::endl;
107    }
108
109   return 0;
110 }
```

The following example allows a user to set to and get data from a joint.

```
1  #include "youbot/YouBotBase.hpp"
2  #include "youbot/YouBotManipulator.hpp"
3
4  using namespace youbot;
5
6  int main() {
7    try {
8      YouBotManipulator myYouBotManipulator("youbot-manipulator");
9      YouBotBase myYouBotBase("youbot-base");
10
11     //command base joint 1 a velocity of 2 radian per second
12     JointVelocitySetpoint setVel;
13     setVel.angularVelocity = 2 *radian_per_second;
14     myYouBotBase.getBaseJoint(1).setData(setVel);
15     setVel.angularVelocity = 0 *radian_per_second;
16     myYouBotBase.getBaseJoint(1).setData(setVel);
17
18     //receive motor current form joint 1 of the manipulator
```

```
    JointSensedCurrent current;
    myYouBotManipulator.getArmJoint(1).getData(current);
    std::cout << "Current manipulator joint 1: " << current.current << std::endl;

    //configure 2 radian_per_second as the maximum positioning speed
    //of the manipulator joint 1
    MaximumPositioningSpeed maxPositioningSpeed;
    maxPositioningSpeed.setParameter(2 * radian_per_second);
    myYouBotManipulator.getArmJoint(1).setConfigurationParameter(maxPositioningSpeed);

  } catch (std::exception& e) {
    std::cout << e.what() << std::endl;
  } catch (...) {
    std::cout << "unhandled exception" << std::endl;
  }
  return 0;
}
```

7.2.2 Keyboard Remote Control

The following source code shows a slightly more complex example involving the KUKA youBot API. The example uses *ncurses* library to get access to the keyboard.

```
#include <iostream>
#include <vector>
#include <signal.h>
#include <ncurses.h>

#include "youbot/YouBotBase.hpp"

using namespace std;
using namespace youbot;

bool running = true;

void sigintHandler(int signal) {
  running = false;
  printf("End!\n\r");
}

int main() {

  signal(SIGINT, sigintHandler);

  try {

    rude::Config configfile;

    if (!configfile.load("../config/applications.cfg"))
      throw ExceptionOODL("../config/applications.cfg file no found");

    int ch = 0;
    double linearVel = 0.05; //meter_per_second
    double angularVel = 0.2; //radian_per_second
    configfile.setSection("KeyboardRemoteContol");
    linearVel = configfile.getDoubleValue("TanslationalVelocity_[meter_per_second]");
    angularVel = configfile.getDoubleValue("RotationalVelocity_[radian_per_second]");

```

```
36      YouBotBase myYouBotBase("youbot-base");
37
38      JointVelocitySetpoint setVel;
39      quantity<si::velocity> longitudinalVelocity = 0 * meter_per_second;
40      quantity<si::velocity> transversalVelocity = 0 * meter_per_second;
41      quantity<si::angular_velocity> angularVelocity = 0 * radian_per_second;
42
43      (void) initscr(); /* initialize the curses library */
44      keypad(stdscr, TRUE); /* enable keyboard mapping */
45      (void) nonl(); /* tell curses not to do NL->CR/NL on output */
46      (void) cbreak(); /* take input chars one at a time, no wait for \n */
47      // (void) echo(); /* echo input - in color */
48
49      def_prog_mode();
50
51      refresh();
52      printf("up = drive forward\n\r"
53              "down = drive backward\n\r"
54              "left = drive left\n\r"
55              "right = drive right\n\r"
56              "y = turn right\n\r"
57              "x = turn left\n\r"
58              "any other key = stop\n\r\n");
59      refresh();
60
61      while (running) {
62
63        ch = getch();
64
65        switch (ch) {
66          case KEY_DOWN:
67            longitudinalVelocity = -linearVel * meter_per_second;
68            transversalVelocity = 0 * meter_per_second;
69            angularVelocity = 0 * radian_per_second;
70            LOG(info) << "drive backward";
71            printf("\r");
72            break;
73          case KEY_UP:
74            longitudinalVelocity = linearVel * meter_per_second;
75            transversalVelocity = 0 * meter_per_second;
76            angularVelocity = 0 * radian_per_second;
77            LOG(info) << "drive forward";
78            printf("\r");
79            break;
80          case KEY_LEFT:
81            transversalVelocity = linearVel * meter_per_second;
82            longitudinalVelocity = 0 * meter_per_second;
83            angularVelocity = 0 * radian_per_second;
84            LOG(info) << "drive left";
85            printf("\r");
86            break;
87          case KEY_RIGHT:
88            transversalVelocity = -linearVel * meter_per_second;
89            longitudinalVelocity = 0 * meter_per_second;
90            angularVelocity = 0 * radian_per_second;
91            LOG(info) << "drive right";
92            printf("\r");
```

```
93            break;
94          case 'y':
95            angularVelocity = angularVel * radian_per_second;
96            transversalVelocity = 0 * meter_per_second;
97            longitudinalVelocity = 0 * meter_per_second;
98            LOG(info) << "turn right";
99            printf("\r");
100           break;
101         case 'x':
102           angularVelocity = -angularVel * radian_per_second;
103           transversalVelocity = 0 * meter_per_second;
104           longitudinalVelocity = 0 * meter_per_second;
105           LOG(info) << "turn left";
106           printf("\r");
107           break;
108
109         default:
110           longitudinalVelocity = 0 * meter_per_second;
111           transversalVelocity = 0 * meter_per_second;
112           angularVelocity = 0 * radian_per_second;
113           LOG(info) << "stop";
114           printf("\r");
115           break;
116       }
117
118       myYouBotBase.setBaseVelocity(longitudinalVelocity, transversalVelocity, angularVelocity);
119
120       refresh();
121       SLEEP_MILLISEC(100);
122     }
123
124     setVel.angularVelocity = 0 * radian_per_second;
125     myYouBotBase.getBaseJoint(1).setData(setVel);
126     myYouBotBase.getBaseJoint(2).setData(setVel);
127     myYouBotBase.getBaseJoint(3).setData(setVel);
128     myYouBotBase.getBaseJoint(4).setData(setVel);
129
130     endwin();
131     SLEEP_MILLISEC(500);
132
133   } catch (std::exception& e) {
134     std::cout << e.what() << std::endl;
135   } catch (...) {
136     std::cout << "unhandled exception" << std::endl;
137   }
138
139   return 0;
140 }
```

Lines 13-18 define a signal handler (this is part of `ncurses` library). This is required when the user wants to interrupt execution through `Ctrl-C` on the command line. Lines 24-29 define an application configuration file which contains configuration options for the build application. This can be found under `~/pkgmanager/openrobots/config`. Lines 30-36 read values for the variables from the section "KeyboardRemoteControl". Lines 37-43 instantiate an object of the KUKA youBot mobile platform with initial linear and angular velocities of zero. Here `quantity` represents physical quantities and their units (lines 40-42). In lines 44-48 `ncurses` library runtime is initialized. Lines 63-118 define a set of statements which are executed when a particular keyboard key is pressed. For instance, when `KEY_RIGHT`

is pressed the lateral velocity of the platform is increased and other velocity components are set to zero. From line 120 to 125 the velocity changes, which were commanded by the switch case, are send to motor controller.

Chapter 8

Robot software frameworks

In addition to the standard drivers and package management software the Live-USB stick that is delivered with the KUKA youBot includes the robotics software framework ROS[1]. Other frameworks such as OROCOS or URBI will follow soon.

8.1 ROS on the KUKA youBot

ROS (Robot Operating System) is a component-based software framework for robotic applications. So-called *ROS nodes* communicate with each other via *messages* and *services*. The formats of these messages are defined in special text files, and can be seen as standardized datatypes in the ROS environment. ROS has a set of tools that make programming easier, for example the *rosmake* tool is able to resolve dependencies to other ROS modules. The build process is also able to automatically transfer the *message* definitions into source code. The algorithms and functionality can be found in different repositories all in all offering a huge amount of functionality. Software is organized in *packages*, which are grouped thematically into *stacks*.

8.1.1 ROS installation on the KUKA youBot

ROS can be used as one framework to operate the KUKA youBot. Currently only ROS Electric and ROS Fuerte are officially supported. The KUKA youBot's ISO image comes with an installation of ROS Electric on Ubuntu 10.04. The installation follows the standard way of installing ROS as described at [14], with a few extensions to add youBot specific packages.
If you want to reinstall ROS Electric on Ubuntu 10.04 follow these steps:

```
$ sudo sh -c 'echo "deb http://code.ros.org/packages/ros/ubuntu lucid main" \
 > /etc/apt/sources.list.d/ros-latest.list'
$ wget http://code.ros.org/packages/ros.key -O - | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install ros-electric-desktop-full
$ sudo apt-get install ros-electric-pr2-controllers
$ echo "source /opt/ros/electric/setup.bash" >> ~/.bashrc
$ echo "export ROS_PACKAGE_PATH=\$ROS_PACKAGE_PATH:/home/youbot/youbot_driver:\
  /home/youbot/ros_stacks:/home/youbot/applications" >> ~/.bashrc
$ . ~/.bashrc
```

With the above operations ROS will be installed to the default directory `/opt/ros/electric`. In order to let ROS find new stacks the environment variable `$ROS_PACKAGE_PATH` has to be set accordingly. Usually this is done either in the file `/opt/ros/electric/setup.sh` or in `~/.bashrc`, here we prefer the second option.

The second last line makes the youBot driver (`/home/youbot/youbot_driver`), the ROS wrapper (`/home/youbot/ros_stacks`) and applications (`/home/youbot/applications`) known to ROS. Actually only the ROS wrapper does depend on ROS, the other two use only rosmake for simpler installation.

Note that all three packages must have been downloaded (resp. the folders must exist), otherwise commands such as roscd will give error messages. If you do not want to use any of the three packages, you should also omit the corresponding entry for the `ROS_PACKAGE_PATH` variable.

8.1.2 Installation of the youBot ROS wrapper

The ROS wrapper allows to write ROS programs for controlling the KUKA youBot. It will be explained in more details in the next section. The ROS wrapper is part of the default installation on the Live-USB stick, that means the following steps are only needed if you want to install it on a different system or for another user.

Before installing the ROS wrapper, the youbot driver must have been intalled as explained in Section 5.2.1. For downloading and compiling the ROS wrapper for ROS Electric enter

```
$ mkdir ~/ros_stacks                                    # (if not done already)
$ cd ~/ros_stacks
$ git clone git://github.com/ipa320/cob_common.git      # brics_actuator messages
$ git clone git://github.com/youbot/youbot-ros-pkg.git
$ cd youbot-ros-pkg                                     # use branch for electric
$ git checkout electric
```

Your system should be working properly if the command `roscd youbot_oodl` goes to the `youbot_oodl` folder. To compile the wrapper for the KUKA youBot API use `rosmake`:

```
$ rosmake --rosdep-install youbot_oodl
```

NOTE: During compilation a window will appear and you will be asked for the administrator password. This is because the driver needs these privileges to be able to send and receive EtherCAT messages. For convenience these privileges are granted at compile time.

In case you get the following error, make sure the X-Window system is working properly. For example if you remotely login to a KUKA youBot via ssh shell please use the -X option:

```
(gksu:12185): Gtk-WARNING **: cannot open display:
make[2]: *** [../bin/youbot_oodl] Error 1
make[1]: *** [CMakeFiles/youbot_oodl.dir/all] Error 2
make: *** [all] Error 2
```

8.1.3 ROS wrapper for the KUKA youBot API

With the `youbot_oodl` driver you can move the base and the arm of the KUKA youBot. You need to send ROS messages as commands. The proprioceptive sensors measurements like odometry and joint angles of the arm and the wheels are published as well. The KUKA youBot API wrapper provides a mapping between the ROS messages and KUKA youBot (OODL) API method calls. An overview of the provided ROS messages is depicted in Figure 8.1.

The messages used for the driver reflect different **levels of abstraction**. One level is the *joint level*, where joint positions in [RAD] and joint velocities in [RAD/s] can be set or received. The mobile platform as well as the arm periodically publish `JointState` messages for position and velocities of the wheels and arm joints. The arm receives `JointPositions` as well as `JointVelocities` messages as commands to either set new joint angles or joint velocities. When you send messages to the arm, you have to use the correct names, otherwise the message is ignored. The joint numbers and the corresponding joint names are listed in Table 8.1.

The mobile base also provides messages on the *kinematic level*. You can send a `Twist` message where you specify the desired translational velocities in [m/s] and rotational velocity in [RAD/s]. The KUKA youBot has an omnidirectional base, so you can set translational values for x and y. The rotation is around the z axis, whereas positive values result in a counterclockwise rotation (see Section 2.1). As part of the *kinematic level* odometry measurements are published as `Odometry` and `tfMessage` frame messages.

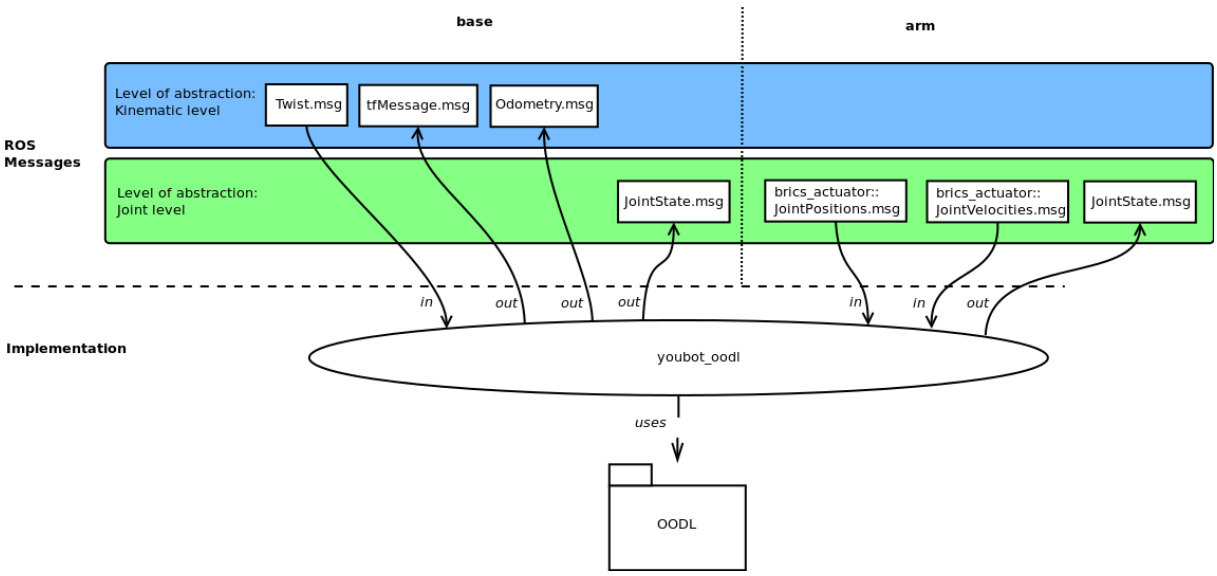

Figure 8.1: Integration of KUKA youBot (OODL) API into the ROS framework.

Joint	Joint name
Wheel #1	wheel_joint_fl
Wheel #2	wheel_joint_fr
Wheel #3	wheel_joint_bl
Wheel #4	wheel_joint_br
Arm joint #1	arm_joint_1
Arm joint #2	arm_joint_2
Arm joint #3	arm_joint_3
Arm joint #4	arm_joint_4
Arm joint #5	arm_joint_5
Joint for the left gripper "finger"	gripper_finger_joint_l
Joint for the right gripper "finger"	gripper_finger_joint_r

Table 8.1: Joint names used for the ROS messages.

8.1.4 Using the ROS wrapper for the KUKA youBot API

To start the driver you have to open a terminal window and type:

```
$ roslaunch youbot_oodl youbot_oodl_driver.launch
```

Per default the driver will assume a KUKA youBot base in combination with one arm. In case only the base or only the arm is available the wrapper will print according error messages but continue to work with the components that are found, as long as proper configuration files are provided as described in Section 5.2.2.

In case you want to visualize the robot and its movements e.g. with the *rviz* tool, open another terminal and type (see also Section 8.1.5):

```
$ roslaunch youbot_oodl youbot_joint_state_publisher.launch
```

The `youbot_oodl` package comes with two simple example applications. The first one is a keyboard based teleoperation application for the base. You can start it with

```
$ rosrun youbot_oodl youbot_keyboard_teleop.py
```

The following keys can be used to move the KUKA youBot,

```
u    i    o
j    k    l
m    ,    .
```

where <i> means forward, <,> backward, <j> left, <l> right, <m> turn left on spot, <.> turn right on spot, <u> move along an arc to the left and <o> move along an arc to the right. All other keystrokes will stop the KUKA youBot.

NOTE: The key bindings work best with a German keyboard layout. In case of a different layout you might consider to adapt the source code.

The second sample application allows to set joint angles for an arm. It asks for values for each arm. When all values are specified, the command will be executed. You can start the sample with:

```
$ rosrun youbot_oodl youbot_arm_test
```

If you want to develop your own application, you have to send messages to the driver, like for example a `JointPositions` message for the arm. The following listing shows an example how to use such messages. In this case the value 0 is send to all joints. Invalid joint values, that exceed the rotational limits will be indicated by a warning.

```
1  static const int numberOfArmJoints = 5;
2  std::stringstream jointName;
3
4  brics_actuator::JointPositions command;
5  vector <brics_actuator::JointValue> armJointPositions;
6  armJointPositions.resize(numberOfArmJoints);
7
8  for (int i = 0; i < numberOfArmJoints; ++i) {
9  jointName.str("");
10 jointName << "arm_joint_" << (i + 1); // Assamble "arm_joint_1", "arm_joint_2", ...
11
12 armJointPositions[i].joint_uri = jointName.str();
13 armJointPositions[i].value = 0; // Here we set the new value.
14 armJointPositions[i].unit = boost::units::to_string(boost::units::si::radians); // Set unit.
15
16 };
17
18 command.positions = armJointPositions;
19 armPositionsPublisher.publish(command);
```

8.1.5 URDF model of the KUKA youBot

The integration of ROS on the KUKA youBot includes a model of the youBot in the **Unified Robot Description Format (URDF)**. This model is a XML file that describes the geometric elements and kinematic chains. It can be visualized with the `rviz` tool. You can either just display the (static) description or you can see the state that is based on the published data from the `youbot_oodl` driver. Figure 8.2 illustrates the visualized URDF model of the KUKA youBot. Before working with the model you have to compile the `youbot_description` package:

```
$ rosmake youbot_description
```

If you just want to see the model, then execute the steps described below, each in a separated terminal:

```
$ roscore
$ rosrun youbot_description joint_robot_publisher
$ roslaunch youbot_description youbot_description.launch
$ rosrun rviz rviz
```

If you want to see the the behavior of the real KUKA youBot start the `youbot_oodl` driver as described above, then start each command in a separated terminal:

```
$ roslaunch youbot_oodl youbot_joint_state_publisher.launch
$ rosrun rviz rviz
```

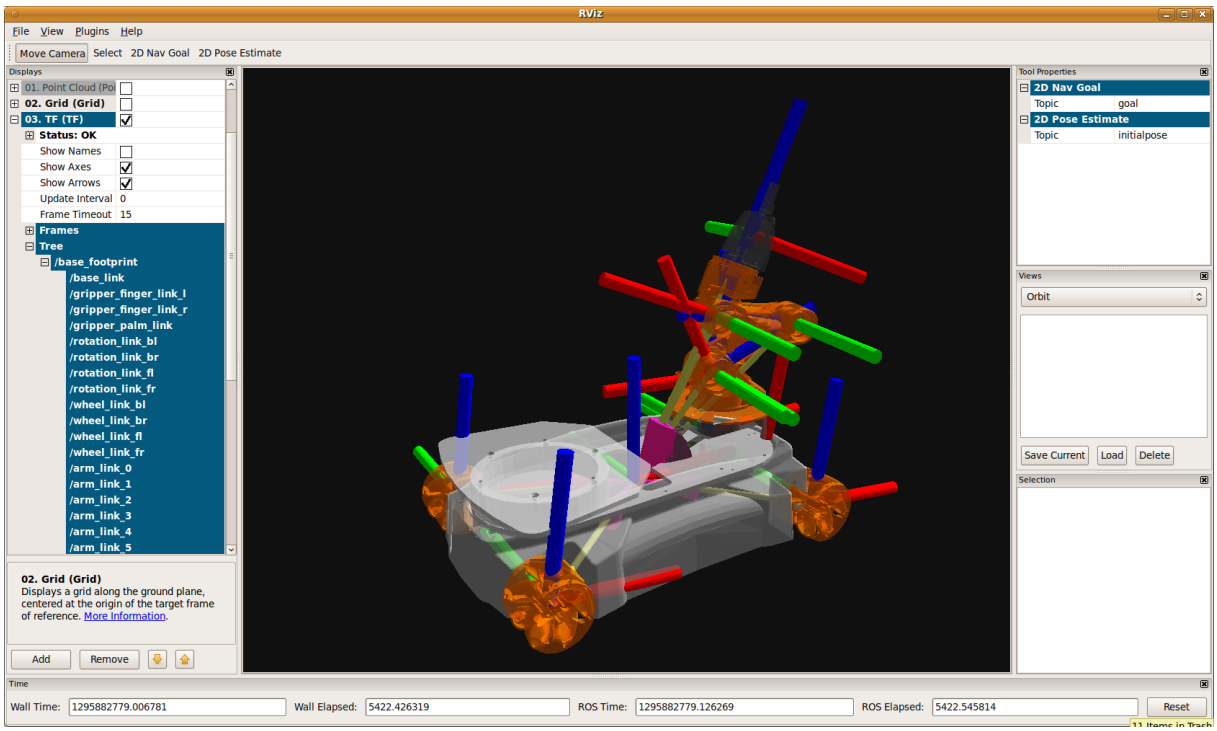

Figure 8.2: The robot model visualized in the `rviz` tool.

Joint	Joint name
Arm joint #1 for 2^{nd} arm	arm_2_joint_1
Arm joint #2 for 2^{nd} arm	arm_2_joint_2
Arm joint #3 for 2^{nd} arm	arm_2_joint_3
Arm joint #4 for 2^{nd} arm	arm_2_joint_4
Arm joint #5 for 2^{nd} arm	arm_2_joint_5
Joint for the left gripper "finger" for 2^{nd} arm	gripper_2_finger_joint_l
Joint for the right gripper "finger" for 2^{nd} arm	gripper_2_finger_joint_r

Table 8.2: Joint names used for the second arm of a dual arm system.

8.1.6 Using a dual arm system

To start the ROS wrapper for a dual arm system first make sure the configuration files for the KUKA youBot are properly setup as described in Section 5.2.2. Then use the following launch script:

```
$ roslaunch youbot_oodl youbot_dual_arm_oodl_driver.launch
```

It will start the driver configured for two arms, load the proper URDF model and start the corresponding `robot_state_publishers`. The two arms have separated topics to receive commands. The first arm accepts messages on the topics that start with the prefix /arm_1/ while commands for the second arm have to be sent to topics with the prefix /arm_2/. Joint names need to be unique to work correctly in conjunction with a URDF file. Thus the joint names of the second arm have new joint names as shown in Table 8.2. The joint names of the first arm remain the same as for a single arm system (cf. Table 8.1). You can use the following example application to send commands to the second arm:

```
$ rosrun youbot_oodl youbot_2nd_arm_test
```

Please note, that /arm_1/ is also the default name for a single arm system. That means applications that make use of a single arm system or the first arm of a dual arm system are compatible.

8.1.7 Simulation in Gazebo

The above described URDF model of the KUKA youBot can be used to simulate the robot in the Gazebo simulator. Currently only the arm is available in simulation. To compile the simulation issue the following commands:

```
$ sudo apt-get install ros-electric-pr2-simulator      # required for simulated youBot
$ rosmake youbot_description
```

To start the simulation of the KUKA youBot arm send the commands below, each in a separated terminal window:

```
$ roslaunch youbot_description youbot_arm_publisher.launch
$ rosrun rviz rviz
$ rosrun youbot_oodl arm_joint_position_control_test
```

You can use the `youbot_arm_test` application to move the arm as already described in Section 8.1.4.

Chapter 9

Safety

9.1 General safety regulations

The following general safety regulations apply to all KUKA youBot components and configurations (in the sequel also called "machine"). That includes the KUKA youBot arm, the omni-directional platform, the platform with arm, and the platform with the two arm system. The same holds for the specific safety regulations, described further below.

- Any KUKA youBot configuration may be taken into operation only if it is in perfect technical condition. Infringements of this instruction may harm humans or animals and damage objects.
- The operator must perform a risk analysis for the machine and convey the safety precautions elaborated in this analysis to the user.
- The operator must analyze and define, which tools and work pieces may be used in an application.
- The operator must perform a risk analysis for the tools of other manufacturer, which (s)he intends to use with the machine.
- It is not allowed to attach any sharp-edged, acuate, or otherwise dangerous items to the machine.
- The machine must be operated only in specified environmental conditions.
- The operator must instruct the user bearing in mind all available risk analyses and considerations.
- The operator must analyze, if the machine has to be equipped with an emergency stop; if the youBot arm is operated with an external power supply, the power supply can be interrupted for an emergency stop.
- During nominal operation neither the operator nor any third party must stay in the danger zone.
- To avoid bodily injuries body parts such as fingers, hands, feet, or head must not be brought into the proximity of the machine. In particular, the head of the user must stay outside of the reachable work space of the machine.
- Electrical and electronic components must not be touched.
- In case (s)he modifies the mechanics or electronics of the machine the operator has to perform a risk analysis of the modified parts. The electronics for charging the battery must not be modified.
- The following threats are possible:
  - Risk of injury after failure of power supply; failure of power supply may disable safety precautions and stop functions; since the youBot arm does not have any motor breaks links and joints may continue to move after a failure of power supply due to kinetic energy stored in the system and to gravitation force.
  - Threats through manipulation of the system.

  - Fire hazard due to overheating, fire caused by shortcuts or overloading of components.
  - Electric shock or electrocution due to missing or damaged insulation.
  - Interference with medical implant of the user through low-frequencies, radio frequencies, or micro waves; observe EMC directives.
  - Risk of injury due to electrostatic processes.
  - Threats due to user or software failures

9.2 Specific safety regulations for KUKA youBot arm

The following safety regulations hold for the youBot arm regardless of wether the arm is operated stationary or mounted on a mobile platform.

- The KUKA youBot arm must be mounted on an appropriate plate so that it cannot overturn or slip.
- The operator has to define the danger zone / work space of the youBot arm.
- Since the limitation of the electrical power to 80W is not implemented in safe technology the operator has to perform a risk analysis which does not rely on this power limitation. The master board of the arm is secured to 12 A and 24 V (288 W). The operator has to establish safety precautions, which reduce the risk to such a level that nobody can be injured and no damage can occur.
- The following threats are possible:
  - Risk of injury due to overturning if the arm is not mounted to a plate.
  - Danger of crushing body parts such as finders during operation.
  - Risk of stitch or cut injuries caused by acuate or sharp tools.
  - Entanglement of hairs or clothing during operation (use hairnet and tight clothing).
  - Risk of injury cause by falling objects or by rapid changes of direction or stops.
  - Threats caused by unexpected motion and changes of direction; if mounted on a mobile robot velocities and threats by unexpected motions of mobile platform and arm add up.

9.3 Specific safety regulations for KUKA youBot omni-directional mobile platform

The following safety regulations hold for the youBot omni-directional mobile platform regardless of wether it is equipped with one or several arms and operated as a so-called mobile manipulator.

- The operator has to perform a risk analysis for the chosen velocity and pay load. For the payload the following regulation has to be obeyed: the payload must be secured, not be dangerous and not exceed the maximum payload.
- The platform must not be operated on a table or some elevated plane of surface.
- The platform may only be operated on horizontal, dry, non-fluffy planes; platform must be protected against accidental fall by suitable arrester barriers; when operated on a tilted the platform may loose traction, slide, and crash.
- While the platform is in motion/operation the user and other persons have to stay outside of its danger zone/workspace.
- The following threats are possible:
  - A risk of injury is caused by rotating wheels; it is recommended to wear compact and closed shoes; it is further recommended to wear hairnets and tight cloths to avoid entanglement of hairs and cloths into wheels.

– Omni-directional platforms can change the direction and velocity of their motion in an unexpected manner and thereby endanger humans, animals or objects in their proximity.
– When the platform is operated in confined space and in the proximity of humans there is a high risk of hits and crushes; wearing compact and closed shoes is recommended.
– A risk of injury can be caused by objects which fall off the robot during operation as a consequence of rapid changes of locomotion direction or sudden stops.
– Fire hazard is caused by potential generation of heat, fire due to shortcuts, and overloading of components.
– Interference with medical implant of the user through low-frequencies, radio frequencies, or micro waves; observe EMC directives.
– Risk of injury due to failure of power supply; failure of power supply may disable safety precautions and stop functions.

Chapter 10

Service and support

The first point of contact for any technical questions and requests for support is:

KUKA youBot Hotline:
(operated by Locomotec UG)

Phone:	+49 1805 968-268
email:	hotline@youbot-store.com
Mailinglist:	users@youbot-community.org
URL:	www.youbot-store.com

In a warranty case, please, contact us first and do not ship back the KUKA youBot yourself. We will take care of shipping and customs issues. Your KUKA youBot will be picked up by our express agent and shipped to our address:

Locomotec UG
Werner-von-Siemens-Str. 6
86159 Augsburg
Germany

Any essential information regarding the KUKA youBot will, of course, be directly communicated to the customers. We still recommend KUKA youBot users and developers to join the users mailing list at `users@youbot-community.org`. We will establish this mailing list as the communication platform for KUKA youBot users. We are also in the process of setting up a wiki for KUKA youBot users under `www.youbot-community.org`.

You may find additional useful information under the following links:

- `http://youbot-store.com`
- `http://youbot-store.com/ybusers.aspx`
- `http://www.facebook.com/KUKAyouBot`
- `http://twitter.com/KUKAyouBot`
- `http://www.youtube.com/KUKAyouBot`
- `http://www.flickr.com/photos/kukayoubot/`
- `http://picasaweb.google.com/kukayoubot`

Bibliography

[1] ROS, "Documentation - ROS Wiki," January 2011. [Online]. Available: http://www.ros.org/wiki/

[2] P. Soetens, *The OROCOS Component Builder's Manual*, 1st ed., 2007. [Online]. Available: http://www.orocos.org/stable/documentation/rtt/

[3] "Directive 2006/42/EC of the European Parliament And Of The Council," Mai 2006. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/site/en/oj/2006/l_157/l_15720060609en00240086.pdf

[4] KUKA, *KUKA youBot Hardware Manual*, January 2011.

[5] Advantech, *AIMB-212 Datasheet*, January 2011. [Online]. Available: http://origindownload.advantech.com/ProductFile/1-F3PAPQ/AIMB-212_DS.pdf

[6] Community Ubuntu Documentation, "GraphicalInstall," February 2011. [Online]. Available: https://help.ubuntu.com/community/GraphicalInstall

[7] ——, "Installation/FromUSBStick," February 2011. [Online]. Available: https://help.ubuntu.com/community/Installation/FromUSBStick#Creating%20a%20bootable%20Ubuntu%20USB%20flash%20drive

[8] UNetbootin, "Homepage and downloads," February 2011. [Online]. Available: http://unetbootin.sourceforge.net/

[9] M. Rostan, J. E. Stubbs, and D. Dzilno, "Ethercat-enabled advanced control architecture," in *IEEE/SEMI Advanced Semiconductors Manufacturing Conference*, July 2010.

[10] SOEM, *SOEM ethercat master*, January 2011. [Online]. Available: http://soem.berlios.de/

[11] Point Grey, "Bumblebee2 CCD FireWire Camera," January 2011. [Online]. Available: http://www.ptgrey.com/products/bumblebee2/bumblebee2_stereo_camera.asp

[12] Mesa Imaging, *SR4000 User Manual*, January 2011. [Online]. Available: http://www.mesa-imaging.ch/dlm.php?fname=customer/Customer_CD/SR4000_Manual.pdf

[13] J. Paulus, *youBot Driver version 0.1*, December 2010. [Online]. Available: http://brics.inf.h-brs.de/youBotApi/

[14] ROS, *Ubuntu install of Electric*, January 2011. [Online]. Available: http://www.ros.org/wiki/electric/Installation/Ubuntu