NIFTi-Arm Quick Manual

Thorsten Linder

Fraunhover IAIS, St. Augustin
⟨thorsten.linder@iais.fraunhofer.de⟩

Project, project Id:	EU FP7 NIFTi / ICT-247870
Project start date:	Jan 1 2010 (48 months)
Actual submission date:	4. März 2013
Lead partner:	Fraunhofer
Revision:	internal

Here comes the abstract

1 The Arm

1.1 Introduction

The NIFTi arm is an extension "module" for the NIFTi-UGV. It can be mounted as replacement of the battery-cover on top of the robot. The arm has 4 degree of freedom. The two main actuators are lifting upper and lower joint to raise a hight up to ca. 1,20 m in addition to the robot hide. On top of the lower joint is a pan-till-unit (PTU) which can carry additional sensors and swivel those into the direction of choice, e.g. cameras. To support an easy replacement of sensors the PTU has 3 active USB 2.0 port available.

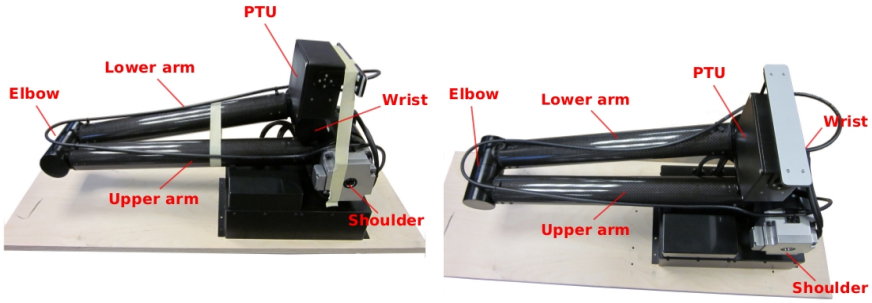

Figure 1: The NIFTi-Arm is composed out of six elements. Starting from bottom up: shoulder, upper arm, elbow, lower arm, wrist and Pan-Tilt-Unit (PTU)

1.2 Characteristics

- Power supply: 24 V DC
- Maximum Height: ca. 120 cm (without robot/Absolem height)
- Maximum Payload: 500 g
- Main Arm Interfaces: CAN-to-USB 2.0
- PTU Interfaces: UTAR-to-USB 2.0
- PTU USB-Hub: 3× USB 2.0
- Mounting: As replacement of the battery-cover (Absolem only)

2 Interfacing

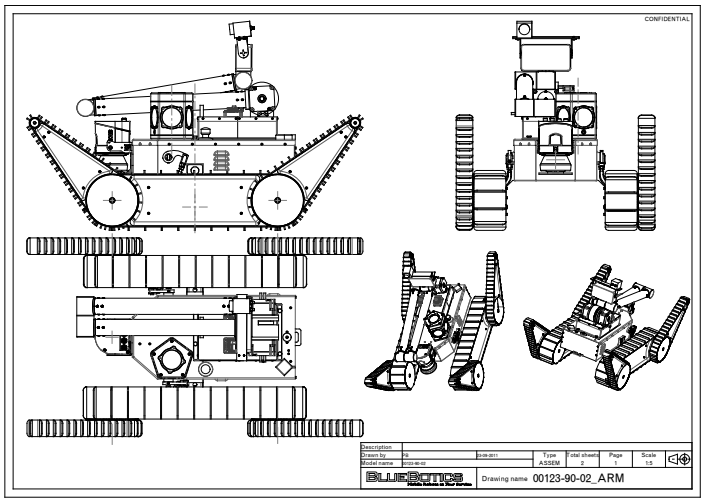

Figure 2: CAD sketch showing the NIFTi-UGV equipped with the arm

The Arm can be mounted as replacement of the battery-cover on top of the robot using the four battery cover bolts.

The power supply is provided by the NIFTi-Robot (check the manual of BlueBotics Absolem for more information). The Arm comes with a short power cable extender which needs to be mounted to the power supply board of the robot. ***!!!Warning!!! Check for the correct polarity !!!Warning!!!***

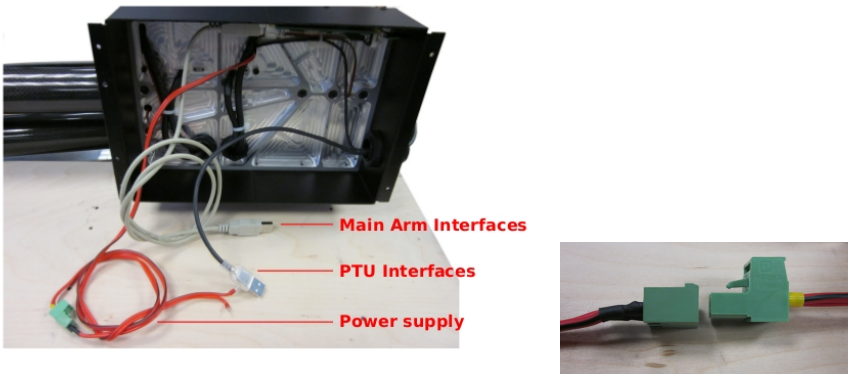

Figure 3: The NIFTi-Arm has three interfaces. The 24V DC power supply interface, a main arm interfaces (CAN-Controller Interface) and a PTU Interface (PTU-controller & PTU USB-Hub)

Furthermore, the main arm interface (USB) and the PTU interface

(USB) need to be plugged to the main board. For further configurations see also 3.1.

Note: The Arm is not part of the safety circuit from Absolem. The power supply board does not have a port which can be controlled via the safety circuits. This means the arm can ***NOT*** *be stopped by the safety switch of the robot.*

3 ROS-Nodes & Usage

The needed Software is available in the NIFTi Subversion at `https://subversion.dfki.de/nifti`

3.1 First Time activation

1. Check-out the NIFTi SVN. In particular you need these packages:
   - nifti_arm (in nifti_drivers)
   - nifti_arm_msgs (in nifti_drivers)
   - nifti_arm_demo_gui (optional; in nifti_ui)
   - drivers (optional; in code)
2. Driver installation; in cases where the drivers are not already installed (e.g. on a laptop or a fresh installed robot) you need to follow the instructions from the README files of the driver (located in the folder driver; not nifti_driver). In particular the CAN-drivers need to be installed. For this purpose just type:

   $ cd cdkl-2.09
   $ sudo make clean install
3. Compile the NIFTi-arm packages

   $ rosmake nifti_arm nifti_arm_demo_gui
4. Creating the udev-rule for the Arm CAN-Controller and the Absolem CAN-Contorller

   By adding the NIFTi-Arm on top of the NIFTi-Robot (Absolem) the System has now two CAN-controllers. The Absolem is configured in such a way that the CAN-controller needs to be enumerated as */dev/usb/cpc_usb0*. The NIFTi-Arm is configured in such a way that the CAN-controller needs to be enumerated as */dev/usb/cpc_usb1*. For more details on *how-to-do* check chapter 5.
5. Check-out and Install ptu motor-drivers

   $ cd ¡any local ros-package-path location¿
   $ mkdir dynamixel_motor && cd dynamixel_motor
   $ svn co `https://ua-ros-pkg.googlecode.com/svn/stacks/dynamixel_motor/trunk`
   (for more information check `http://www.ros.org/wiki/dynamixel_controllers`)
   $ rosmake dynamixel_controllers

6. First time starting
   Ensure the arm is folded together like depict in figure 1, then start the main driver
   $ roslaunch nifti_arm arm.launch
   and start the demonstration GUI
   $ roslaunch nifti_arm_demo_gui node.launch

4 The Start-Up Behaviour

Ensure the arm is folded together like depict in Figure 1 & Figure 4. Be aware that, the arm will move directly after the launch of the arm node ($ roslaunch nifti_arm arm.launch). Ensure that the arm is free to move upwards.

For self calibration purpose the arm will ensure that its limbs are moveable. After that the limbs will be parked in the default position (Figure 1). Therefore, both joints (lower and upper arm) will move a little upwards and immediately return to the folded position. By doing so the joints will release and press again there position end switches attached to the joint parking brackets. On success the arm is now calibrated and ready to use.

In cases where the arm node was launched while the joints were not entirely closed, the arm will also try to open and close till the parking position is reached. Never the less a success in this case is not guaranteed as outer factors like blocking obstacles or violation of maximum payload criteria can not be taken into account.

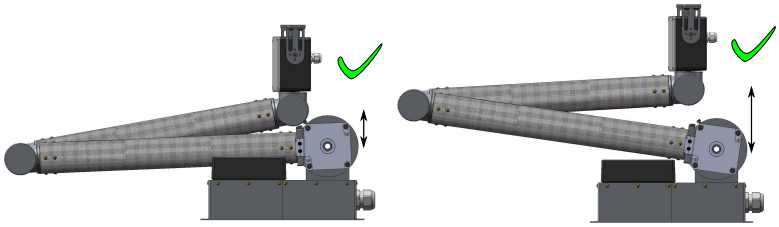

Figure 4: Left: Default resting/parking position; Right: Normal start up lifting height

4.1 The Height Based Control Method

The preferential mode of control is the height based control. In this mode the arm is commanded to rise a certain hight, i.e. the PTU shall be lifted up to the commanded height. In this mode the arm will rise vertically, whereby the PTU is sliding up- or downwards on a virtual vertical line starting from the center of the shoulder [1].

The used ROS message is shown provided in the *nifti_arm_msgs* package and contains only one parameter for the targeting height (cf. Listing 1).

Listing 1: Body of the ROS height control message

```
1 #Header header
2 int16 height #Height in millimeters
```

[1] The model of a virtual vertical line rising from the shoulder does only hold true if the arm is already erected for a few cm. As it can be seen in 4

Figure 5: Arm erected in full height

4.2 The Manual Based Control Method

As an alternative to the height based control method, the arm further supports a manual control method. In this mode the degree of freedoms of the arm can be commanded separately, i.e. the angular orientation of each motor can be commanded by hand.

For keeping the control simple the ROS message *msg_position.msg* is in use, provided by the *nifti_arm_msgs* package and contains two parameters (cf. Listing 2). Parameter *a* is corresponding to the target orientation in radiant of the shoulder motor and parameter b accordingly for the elbow motor. The *arm_listener* node is listening for messages and will move both motors simultaneously to the target orientation.

Listing 2: Body of the ROS position control message

```
1 #Sends the position of the motors in rad
2 #header header
3 float32 a
4 float32 b
```

!!!Warning!!! Using the manual control mode can lead to arm self-collisions which can damage the system. Take precautions to ensure that the commanded movements can not end in a self collision of the arm with it self. Also ensure that the arm can not collide with other robot parts.

Figure 6 shows the most common self collision situation. Will rising or lowering the arm from the resting position the PTU can collide with parts of the wrist.

Be aware that the limbs are producing a certain amount of leverage force. Figure 7 indicates some situation in which the leverage forces are acceptable or must be avoided. In general speaking, the length of the lever from the limbs shall be smaller or equal to the total length of one limb.

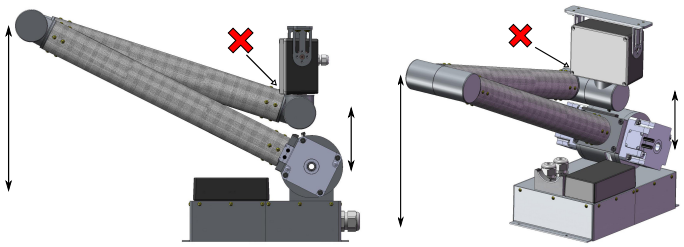

Figure 6: Self-collision of the arm with its wrist.

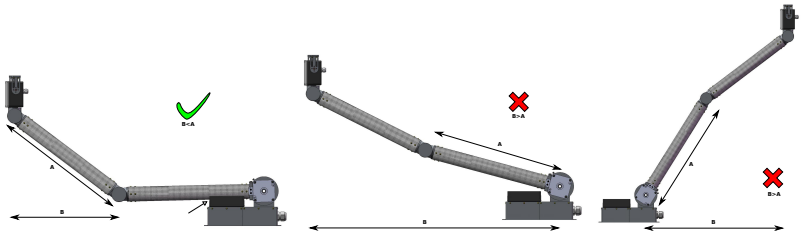

Figure 7: Left: Situation where the leverage force is acceptable. Middle and Right: Situation with to high leverage forces, i.e. the arm can get damaged.

5 Space Device Mapping

udev is the device manager for the Linux kernel. Primarily, it manages device nodes in */dev*, i.e. it manages the mapping from devices to the user space (file-names within the */dev* folder). The user can influence the mapping procedure by rules, i.e. a set of scripts stored in the */etc/udev/rules.d* folder[1].

Fore the usage of the NIFTi-arm several rules need to be available on the system/robot. As the robot and the arm are both using a separated CAN-USB-Interface, the system needs to ensure that the mapping of those two devices is constant. This means the system needs one rule which ensures that the robot CAN-USB-Interface will be mapped to */dev/usb/cpc_0* and another one for the arm which needs to be mapped to */dev/usb/cpc_1*. Currently there is one generic rule for the interfaces installed which will not ensure for all cases the correct mapping. This one rule needs to be deactivated first.

5.1 Disabling the generic robot udev-rule

The default udev-rule for the CAN-USB-Interfaces is generic and does not ensure the needed mapping. There for deactivation of this rule is needed.

[1] see also `http://www.reactivated.net/writing_udev_rules.html` for further information

1. To do so you need to find first the rule on your system.

   $ cd /etc/udev/rules.d
   $ ls -al *-cpc.rules

   The output will look like this one.

```
-rw-r--r-- 1 root root 602 Oct 10 14:05 81-cpc.rules
```

2. To validated that this rule is the original generic one take a look inside.

   $ cat 81-cpc.rules

   The output will look like this one (Listing 3).

Listing 3: The generic rule

```
# Move /dev/cpc_usb* to /dev/usb/cpc_usb*
KERNEL=="cpc_usb*", NAME="usb/cpc_usb%n", MODE="0666"

# /dev/cpc_eco*
KERNEL=="cpc_eco*", NAME="cpc_eco%n", MODE="0666"
KERNEL=="cpc-eco"

# /dev/cpc_xt*
KERNEL=="cpc_xt*", NAME="cpc_xt%n", MODE="0666"
KERNEL=="cpc-xt"

# /dev/cpc_xti*
KERNEL=="cpc_xti*", NAME="cpc_xti%n", MODE="0666"
KERNEL=="cpc-xti"

# /dev/cpc_pp*
KERNEL=="cpc_pp*", NAME="cpc_pp%n", MODE="0666"
KERNEL=="cpc-pp"

# /dev/cpc_pci*
KERNEL=="cpc_pci*", NAME="cpc_pci%n", MODE="0666"
KERNEL=="cpc-pci"

# /dev/cpc_card*
KERNEL=="cpc_card*", NAME="cpc_card%n", MODE="0666"
KERNEL=="cpc-card"
```

3. After validation that the generic rule is installed it needs to be deactivated. For this the file will be renamed, i.e. the number in front of the file name will be removed.

   $ sudo mv 81-cpc.rules cpc.rules.old

4. To remove also the rule from cache udev must now be updated. A reload of the daemon will do the job.

$ sudo service udev reload

5.2 Creation of new udev-rules for the Robot

To create a new rule for the robot which ensures the correct mapping, some informations must first be collected. Then those information will be used to create a dedicated rule for the robot CAN.

1. At first ensure that the arm CAN-USB-Interface is **not** connected to the robot, this will avoid confusion.

2. Each device is sending information about there manufacturer, product-typ and serial number to the udev manager. Reqeusting those information from the manager with the command

   $ udevadm info -a -p $(udevadm info -q path -n /dev/usb/cpc_usb*)

   will return similar data like Listing 4.

Listing 4: The robot CAN-USB udev information

```
1
2  Udevadm info starts with the device specified by the
       devpath and then
3  walks up the chain of parent devices. It prints for
       every device
4  found, all possible attributes in the udev rules key
       format.
5  A rule to match, can be composed by the attributes of
       the device
6  and the attributes from one single parent device.
7
8    looking at device '/devices/pci0000:00/0000:00:1a.0/
         usb1/1-1/1-1.2/1-1.2:1.0/usb/cpc_usb0':
9      KERNEL=="cpc_usb0"
10     SUBSYSTEM=="usb"
11     DRIVER==""
12
13   looking at parent device '/devices/pci0000
         :00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.0':
14     KERNELS=="1-1.2:1.0"
15     SUBSYSTEMS=="usb"
16     DRIVERS=="cpc-usb"
17     ATTRS{bInterfaceNumber}=="00"
18     ATTRS{bAlternateSetting}==" 0"
19     ATTRS{bNumEndpoints}=="03"
20     ATTRS{bInterfaceClass}=="ff"
21     ATTRS{bInterfaceSubClass}=="ff"
22     ATTRS{bInterfaceProtocol}=="ff"
23     ATTRS{supports_autosuspend}=="0"
24     ATTRS{interface}=="Interface 0"
```

```
25
26    looking at parent device '/devices/pci0000
          :00/0000:00:1a.0/usb1/1-1/1-1.2':
27      KERNELS=="1-1.2"
28      SUBSYSTEMS=="usb"
29      DRIVERS=="usb"
30      ATTRS{configuration}==""
31      ATTRS{bNumInterfaces}==" 1"
32      ATTRS{bConfigurationValue}=="1"
33      ATTRS{bmAttributes}=="80"
34      ATTRS{bMaxPower}=="500mA"
35      ATTRS{urbnum}=="15"
36      ATTRS{idVendor}=="12d6"
37      ATTRS{idProduct}=="0444"
38      ATTRS{bcdDevice}=="0100"
39      ATTRS{bDeviceClass}=="ff"
40      ATTRS{bDeviceSubClass}=="ff"
41      ATTRS{bDeviceProtocol}=="ff"
42      ATTRS{bNumConfigurations}=="1"
43      ATTRS{bMaxPacketSize0}=="16"
44      ATTRS{speed}=="12"
45      ATTRS{busnum}=="1"
46      ATTRS{devnum}=="17"
47      ATTRS{devpath}=="1.2"
48      ATTRS{version}==" 1.10"
49      ATTRS{maxchild}=="0"
50      ATTRS{quirks}=="0x0"
51      ATTRS{avoid_reset_quirk}=="0"
52      ATTRS{authorized}=="1"
53      ATTRS{manufacturer}=="EMS Dr. Thomas Wuensche"
54      ATTRS{product}=="CPC-USB"
55      ATTRS{serial}=="0001822"
56
57    looking at parent device '/devices/pci0000
          :00/0000:00:1a.0/usb1/1-1':
58      KERNELS=="1-1"
59      SUBSYSTEMS=="usb"
60      DRIVERS=="usb"
61      ATTRS{configuration}==""
62      ATTRS{bNumInterfaces}==" 1"
63      ATTRS{bConfigurationValue}=="1"
64      ATTRS{bmAttributes}=="e0"
65      ATTRS{bMaxPower}==" 0mA"
66      ATTRS{urbnum}=="368"
67      ATTRS{idVendor}=="8087"
68      ATTRS{idProduct}=="0020"
69      ATTRS{bcdDevice}=="0000"
70      ATTRS{bDeviceClass}=="09"
71      ATTRS{bDeviceSubClass}=="00"
```

```
72      ATTRS{bDeviceProtocol}=="01"
73      ATTRS{bNumConfigurations}=="1"
74      ATTRS{bMaxPacketSize0}=="64"
75      ATTRS{speed}=="480"
76      ATTRS{busnum}=="1"
77      ATTRS{devnum}=="2"
78      ATTRS{devpath}=="1"
79      ATTRS{version}==" 2.00"
80      ATTRS{maxchild}=="6"
81      ATTRS{quirks}=="0x0"
82      ATTRS{avoid_reset_quirk}=="0"
83      ATTRS{authorized}=="1"
```

3. The needed information are contained in the lines starting with the attributes ATTRS{manufacturer}, ATTRS{product} & ATTRS{serial}. Here the attributes are:

```
53      ATTRS{manufacturer}=="EMS Dr. Thomas Wuensche"
54      ATTRS{product}=="CPC-USB"
55      ATTRS{serial}=="0001822"
```

   Those lines will be used for the rule to identify the interface uniquely.

4. Having all needed information a new rule will be create with an arbitrary text editor (here vim).

   $ sudo vim /etc/udev/rules.d/91-cpc-robot.rules

   Insert the command from Listing 5 and store the rule. **Note**: that the serial attribute will be different for you.

Listing 5: The robot rule

```
1 KERNEL=="cpc_usb*",ATTRS{manufacturer}=="EMS Dr. Thomas
      Wuensche",ATTRS{product}=="CPC-USB",ATTRS{serial
    }=="0001822", NAME="usb/cpc_usb0", MODE="0666"
```

5. Now the udev manage needs to be updated. Either restart the robot, or unplug the robot CAN-interface and type

   $ sudo service udev restart

   Then plug-in the CAN-Interface again and verify that the rule is mapping the robot to */dev/usb/cpc_usb0*.

   $ ls /dev/usb/cpc_usb0

5.3 Creation of new udev-rules for the Arm

The procedure of rule creation for the NIFTi-Arm follows according the methods of section 5.2, with a few exceptions.

1. At first the CAN-USB-Interface of the Arm will be connected and the robot shall be disconnected.
2. The rule name for the arm should now be a different from the former one. It is assumed that the arm rule will be named *90-cpc-arm.rules*.
3. Last but not least, in the rule the mapping command *NAME="usb/cpc_usb1"* must be used. This will ensure that the Arm-Interface will be mapped to */dev/usb/cpc_usb1*.

6 Recovery Guide for the Motor Controllers

In cases where the motor controllers need to be reconfigured/recovered or where settings must be changed, you will need to use a windows PC and a suitable serial flashing cable.

At first download the ELMO Composer software (`http://www.elmomc.com/products/software-tools-main.htm`) and the configuration file (take a look at the NIFTi-SVN `File://nifti_drivers/trunk/nifti_arm/recovery-files`).

Remove the motor controller cover (the black plastic box on top of the arm base). Then, for each controller you need to perform the following steps:

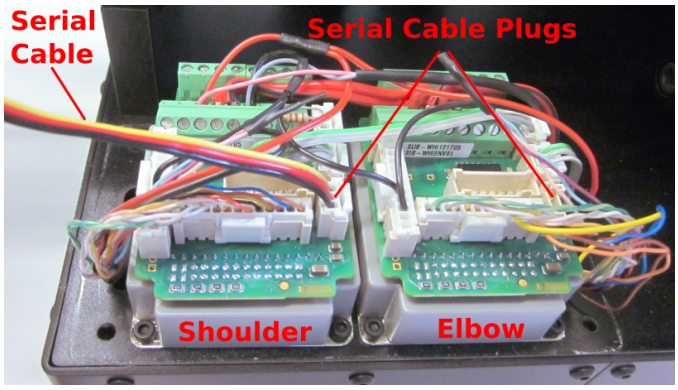

Figure 8: The two main motor controller of the arm. Left: shoulder controller; Right: elbow controller

1. Connect the serial cable to the targeted motor controller (see Figure 8).
2. Use the Composer to flash the motor controller with the configuration provided by the configuration file (.dat file). Ensure to use the correct file for the targeted controller[2].
3. Use the console provided by the Composer software and type

```
PP[13]=<CAN-ID>;
PP[14]=<CAN-BAUD>;
MO=0;
SV;
```

   Whereby the variable <CAN-ID> is 20 for shoulder or 21 for elbow. Whereby the variable <CAN-BAUD> is by default 1, which is equivalent to 500kbit/s.

[2]If you mix up the configuration the motor directions will be inverted, i.e. the arm will malfunction.

4. Check the settings by un-powering the device, waiting for 10 seconds and re-powering. Then type the following in to the Composer console, to see the stored values:

```
PP[13];
PP[14];
```

   The returned values should be equal to the commanded ones.

7 Safety instructions

- Do not use the arm in an erected mode while driving.
- Do not use the arm without line of view or a safety personal. The BlueBotic Absolem robot does not support an emergency switch function for the arm.
- The build-in USB-Hub is actively powered. However it may happens that the supported current is not enough to power all your devices.
- The Arm is designed to lift sensors in a vertical means. It is NOT designed to manipulated objects or to stretch/move the sensors horizontal. This means using the arm in such a way that the center of gravity is not only vertically moved may causes damages to the arm.
- The extra weight of the arm moves the center of gravity of the robot and can unbalance it.
- Do not fall on the arm.
- Do not crash or collide.