FourByThree Ros Control layer

This Document deals with the installation and the stand-alone use of the Control layer of the FourByThree Robot.

In order to install the software, it is recommended to the users to contact the developers by mail (iras@itia.cnr.it or manuel.beschi@itia.cnr.it) to obtain the rosinstall files and information on the required operating systems and news on possible changes on bugfix in the code. The developers staff will provide an open access to the Control layer repositories and the NDLCOM driver software.
It is worth stressing that the repositories are subjected to continuously updates, therefore before each significant code change a new branch named fourbythree_XX.YY will be created for the users that wants to keep unchanged their own code.

Table of Contents

NDLCOM DRIVER - INSTALLATION PROCEDURE

The communication between the ROS suite and the robot is provided by using a NDLCOM driver. In order to install the driver, the user have to follow this procedure:

1) Install required dependency:

sudo apt-add-repository ppa:brightbox/ruby-ng
sudo apt-get update
sudo apt-get install qt4-dev-tools libqt4-dev libqt4-core libqt4-gui cmake libdb5.1++-dev pkgconf libxml++2.6-dev libxml2-utils libffi-dev lzma-dev lzma liblzma-dev libxslt-dev
sudo apt-get install ruby2.2 ruby2.2-dev gem-dev git
sudo gem2.2 install nokogiri

2) Add USB access to user:
```
sudo usermod -a -G dialout $USER
```
sudo reboot

3) Create a workspace (do not use the ROS workspace). Example: *mkdir ndlcom_ws*
3) *git clone https://bitbucket.org/CNR-ITIA/ndlcom_driver* (USERNAME and PASSWORD will be provided by mail)
3) *cd ndlcom_driver*
4) *git checkout communication_problem*

4) *cd ndlcom_driver*
5) *source install_ndlcom.bash*

ROS PACKAGES - INSTALLATION PROCEDURE

This section deals with the FourByThree ROS control layer installation.

NOTE1: In order to use the real robot, you have to install the NDLCOM driver before compiling the ROS part. If the NDLCOM driver is not installed, you can use only simulated configurations. If you have installed the NDLCOM driver after compiling the ROS package, you have to remove the build/ and devel/ folders in the catkin_workspace and compile it again.
NOTE2: If you are going to use the stiffness recommendation functionality, you have to install the execution manager framework (please ask to Tekniker staff for instructions) before compiling. If you have installed the execution manager framework after compiling the ROS package, you have to remove the build/ and devel/ folders in the catkin_workspace and compile it again.

USE THE LAST UPDATED RELEASE

1) Install dependency:
sudo apt-get instal ros-indigo-moveit-full
sudo apt-get instal ros-indigo-ros-control
sudo apt-get install ros-indigo-moveit-simple-controller-manager
sudo apt-get install python-wstool

2) Create our ROS workspace if needed.
3) *cd [PATH_TO_YOUR_WORKSPACE]*
4) wstool init src
5) *wstool merge -t src /PATH/TO/fourbythree.rosinstall*. If you already download a specific package, it will ask you to replace it. USERNAME and PASSWORD will be provided by mail
6) *wstool update -t src*
7) *catkin_make*

UPDATE PROCEDURE

The ROS layer is a live software packages, thus it is continuously updated to improve performance and code stability. In case of serious updates, the users will be invitated to update the source code by using the following procedure.

1) cd [PATH_TO_YOUR_WORKSPACE]
2) wstool update -t src
3) catkin_make

NOTE:

If you modified some files inside the repositories, the update procedure will fail.
To remove the local changes:
1) cd [PATH_TO_YOUR_WORKSPACE]/src/[REPOSITORY]
2) git stash
3) cd [PATH_TO_YOUR_WORKSPACE]
4) wstool update -t src
5) catkin_make

USE THE LAST STABLE RELEASE:

Some users could prefer to work with a freezed version of the code. In that cases, they can switch to the a stable software release by using the following procedure:

1) Install dependency:
sudo apt-get instal ros-indigo-moveit-full
sudo apt-get instal ros-indigo-ros-control
sudo apt-get install ros-indigo-moveit-simple-controller-manager
sudo apt-get install python-wstool

2) Create our ROS workspace if needed.
3) *cd [PATH_TO_YOUR_WORKSPACE]*
4) wstool init src
5) *wstool merge -t src /PATH/TO/fourbythree_stable_XX_YY.rosinstall*. If you already download a specific package, it will ask you to replace it. USERNAME and PASSWORD will be provided by mail
6) *catkin_make*

RUN THE ROS CONTROL LAYER IN STAND-ALONE MODE

To run the full controller, open a terminal and run the following command:

roslaunch fourbythree_ros_controller start_complete_layer.launch configuration:=proto5_gs robot:=$[YOUR_ROBOT_NUMBER] gs:=true

where [*YOUR_ROBOT_NUMBER*] is the prototype number:
1 PROTO1@DFKI: old prototype, 2DOF
2 PROTO2@TEKNIKER: old prototype, 5DOF
3 PROTO3@ITIA: old prototype, 6DOF
4 PROTO4@STODT: new prototype, 6DOF
5 PROTO5@TEKNIKER: new prototype, 6DOF
6 PROTO6@ZEMA: new prototype, 6DOF
7 PROTO7@ZEMA: new prototype, 6DOF

To run the 6DOF basic controller, open a terminal and run the following command:

oslaunch fourbythree_ros_controller start_complete_layer.launch configuration:=basic robot:=$[YOUR_ROBOT_NUMBER]

To run the 5DOF basic controller, open a terminal and run the following command:
oslaunch fourbythree_ros_controller start_complete_layer.launch configuration:=basic robot:=$[YOUR_ROBOT_NUMBER]

To run the 6DOF aimulator, open a terminal and run the following command:
oslaunch fourbythree_ros_controller start_complete_layer.launch configuration:=simulation_planner robot:=$[YOUR_ROBOT_NUMBER]

Configurations description (configuration names could be changed in future release)

- *basic*: basic 6DOF configuration, where only the motor planner and the communication driver are loaded
- *basic_5dof*: basic 5DOF configuration, where only the motor planner and the communication driver are loaded
- *proto5_gs*: 6DOF link-controller configuration with oscillations damping.
  It is possible to communication with the program executor and the handheld.
  oscillation compensation is enable, vertical positions may present oscillations due to play.
  It has to be used with second version 6DOF prototypes.
- *simulation_planner*: simulation mode

NOTE: when you change from one configuration to another one, make sure to restart the roscore.

USE THE ROS CONTROL LAYER

Launcher *start_complete_layer.launch* start the communication with the driver and switch on the robot (if the chosen configuration is not *simulation_planner*) and a *rqt* window with the error/warning monitor.

In the terminal, user can:

- press '**s**' to store the offset in ~/.ros/motor_offset. In the following executions, the program automatically load these values. The offset can change every time you press the emergency stop or you switch off the power supply.
  In this case,
  - manually remove the file ~/.ros/motor_offset
  - restart the control layer
  - store again the offset
  - SUGGESTION: try to switch off the robot (namely to press the emergency button or to turn off the power supply) the minimum number of times. At the contrary, you can stop the controller_layer terminal whenever you want.

- Press '**u**' to unload the driver without stopping the entire framework. The brake will be turned on.
- Press '**l**' to (re)load the driver. Controller will start in *link_controller* mode.
- Press '**x**' to gently unload the driver and stop the nodes.

In case of critical error, the driver is turned off and the break are activated.

Troubleshooting:

Error and warning on the ndlcom driver and on the ros controller are stored in:
/home/USERNAME/.ros/ndlcom_error.txt
/home/USERNAME/.ros/diagnostic_log_DATESTRING.bag

In case of strange errors, please contact manuel.beschi@itia.cnr.it providing these files.
NOTE: old files /home/USERNAME/.ros/diagnostic_log_DATESTRING .bag can delete if you need more space on the disk.

Some common problems are:

- *COMPILING WARNING:*

warning "Cannot compile ndlcom_driver package because the ndlcom_device_driver is not installed."
the compiler is not able to locate the ndlcom library (See **NOTE1**).
User has to reinstall it by following the instructions of Section **NDLCOM DRIVER - INSTALLATION PROCEDURE.**

- If the robot does not move due to a spring offset error, you can use this workaround:

rosservice call /virtual_sensor_hi/virtual_sensor/virtual_sensor_zeroing "{}"
rosservice call /fourbythree/controller_manager/switch_controller "start_controllers:
- 'CONTROLLER_NAME'
stop_controllers:
- ''
strictness: 1"

- *RUNTIME ERROR:*

[ERROR] [1504276129.106933522]: The error before refreshing the cache was: Could not find library corresponding to plugin ndlcom_ros/NdlcomDriverNodelet. Make sure the plugin description XML file has the correct name of the library and that the library actually exists.
if this error happens the first time after compiling, simply reboot the system to refresh the cache.

- *RUNTIME ERROR:*

[ERROR] "[NdlcomDriver::NdlcomDriver] ERROR(Serialcom): UART_connect failed, 'm_port_handle=XXX."
or
[WARN] [1507551500.244302616]: Robot configuring failed
The driver is not able to open the serial port.

Stop the driver, check the port number by using:
ll /dev/ | grep USB
if the port number is not 0 (namely, ttyUSB0), please specify the port by using:

*roslaunch fourbythree_ros_controller start_complete_layer.launch configuration:=proto5_gs robot:=$[YOUR_ROBOT_NUMBER] gs:=true port:=/dev/ttyUSB**X***
If the problem remains, check the power supply, unplug and plug again the USB cable and repeat the procedure.