rapros: User's guide

Luca Cavanini, Gionata Cimini

[a]Dipartimento di Ingegneria dell'Informazione
Università Politecnica delle Marche,
Via Brecce Bianche 12, 60131, Ancona, Italy
{l.cavanini,g.cimini}@univpm.it

1 ROS Environment Configuration

The installation and testing of *rapros* package are realized on embedded boards, namely BeagleBone Black and BeagleBoard-xM, running Linux kernels and the ROS framework. The setup procedure on this low-cost boards is divided into three steps: (I) Linux kernel installation, (II) ROS environment configuration, (III) *rapros* package setup. It is worth noticing that *rapros* package could be tested also on a single PC, running ROS and Simulink.

In the following, the setup and configuration procedure are explained. The guidelines highlight Unix OS commands; obviously the procedure can be performed on Microsoft Windows© systems. On Windows PC, Putty software is required [1]; in this case the correct configuration includes board IP (port 22) and *ssh* connection type.

1.1 BeagleBone Black setup

BeagleBone Black is a low-cost development platform, which provides a AM335x 1GHz ARM® Cortex-A8 with 512MB DDR3 RAM and 3D acceleration.
It runs Ångström, a Linux distribution developed in order to work ARM units. The Ångström kernel image can be download from [2]. The installation of the operating system is easy following the guidelines in [3]. In the following we provide the main steps to build up the complete system. The board communicates with a PC through a USB port and a *ssh* connection, established with:

```
> sshroot@beaglebone.local
```

Usually username and password are both *root* at the first log-in.
Even if it is not mandatory, it is preferable to have an updated version of Ångström:

```
> opkg update
> opkg updrage
```

The ROS version, suitable for the board, can be downloaded from a *git* project. To directly clone the repository into the system, the last software version must be installed through *opkg*:

```
> opkg install git
> git clone git://github.com/vmayoral/beagle-ros.git
```

To install the ROS framework, an easy to use shell script is provided.

```
> cd beagle-ros/scripts
> bash ./minimal-ros-install-angstrom.sh
```

Once the installation is finished, it is advisable to configure the ROS environment variables in the *.bashrc* file. This is possible by using a software editor, like nano, typing in home directory:

```
> nano .bashrc
```

and adding to the end of the opened file, the following raws:

```
export ROS_HOSTNAME=localhost
export ROS_MASTER_URI=http://localhost:11311
```

The workspace is created during the installation task, but if this does not happen, it is possible to create it through to ROS tutorial on the reference website.

1.2 BeagleBoard-xM setup

BeagleBoard-xM offers a AM37x 1GHz ARM Cortex-A8 processor, with 512 MB LPDDR RAM and 3D acceleration. In the following the procedure to install ROS on the board is provided. In this case, the OS selected for the installation is the ARM version of Linux Ubuntu 10.04 LTS. This Linux distribution is more stable then the others Linux OS embedded alternatives. Indeed Ångström release for this board is not still fully working.

The OS is downloaded and copied on the MicroSD card, and some local parameters must be set:

```
> wget https://rcn-ee.net/rootfs/2015-01-06/microsd/bbxm-ubuntu-14.04.1-
console-armhf-2015-01-06-2gb.img.xz
> unxz bbxm-ubuntu-14.04.1-console-armhf-2015-01-06-2gb.img.xz
> sudo dd if=./bbxm-ubuntu-14.04.1-console-armhf-2015-01-06-2gb.img
of=/dev/sdX
> sudo update-locale LANG=C LANGUAGE=C LC_ALL=C LC_MESSAGES=POSIX
```

In this case the ROS framework can be installed following the standard tutorial for desktop installation [4]. Occasionally, the authors have experienced unwanted changes in Ethernet MAC address after a reboot. The following lines are sufficient to avoid the problem:

```
> sudo gedit /etc/network/interfaces
> auto eth0
> iface eth0 inet dhcp
> hwaddress ether 01:02:03:04:05:06 > sudo /etc/init.d/networking restart
```

where the fourth command requires the correct Ethernet MAC address

1.3 *rapros* package installation

rapros package can be downloaded from [link al repository]; the package come with two working examples:

- a customized *rapros* node and the related Simulink block to perform a loop-back test in Simulink and the ROS environment;
- the node and the Simulink block developed during the PIL test.

To install the package, *rapros* must be placed in the /src folder inside the ROS workspace. From MATLAB side, the Matlab_Simulink directory inside the selected example package must be added to the Matlab path. The *rapros* with the Simulink block will appear in the Matlab palette. Finally, each launch file and each *rapros* python file, in the launch directory needs the administrator privileges to be launched. This is possible typing:

```
> sudo chmod +x name_file.launch
> sudo chmod +x name_file.py
```

1.4 *rapros* parameters setup

rapros package requires the correct setup of the connection parameters. These parameters are set on the launch file, for the ROS side, and on the *rapros* block mask, for the Simulink side. The launch file allows the modification of the following paramenters:

- **ip_board**: in the IP parameter group, indicates the IP address of the board connected to the personal computer through direct Ethernet connection;
- **ip_pc**: in the IP parameter group, indicates the IP address of the personal computer connected to the board through direct Ethernet connection;
- $\mathbf{T_s}$: indicates the sample time of the control system.

The user is encouraged to run the loop-back test, to verify the correct behaviour of the system. The example is located in *rapros_test*. It can be completely run on a PC without the use of a board, setting the *ip_board* and *ip_pc* to the local host address, namely 127.0.0.1.

In "*rapros* Simulink block", the parameters can be set through the block mask. The following fields can be customized:

- **Input dimension**: it indicates the dimension of the input from the Simulink to the *rapros* block;

- **Output dimension**: it indicates the dimension of the output from the *rapros* block to the Simulink model;
- **Sample time**: it indicates the sample time of the control system, it must be the same of T_s value set in the launch file;
- **Host address receive**: this parameter indicates the address of the personal computer connected to the board through direct Ethernet connection;
- **Host address send**: it indicates the address of the board connected to the PC through direct Ethernet connection.

Once the configuration parameters are correctly set, the launch file of the needed example package starts the ROS-side package. If everything is working, the following message will be displayed:

```
wait upd packet
```

When the simulation starts in Simulink, the *rapros* node is directly synchronized with the model. If the *rapros* node stops, Simulink model stops the simulation with an error. When Simulink interrupts the simulation, the node returns in wait state. It is worth noticing that after each simulation, the *rapros* node must be restarted, in order to reset the paramter server of ROS Master, and the variables' values inside the node.

References

1. www.putty.org.
2. http://www.angstrom-distribution.org/building-angstrom.
3. http://downloads.angstrom-distribution.org/demo/beaglebone/.
4. http://wiki.ros.org/indigo/Installation/UbuntuARM.