ROS Documentation

Documentation on how to start-up the RPI and ROS master.

Links to Basic ROS tutorials
Core ROS Tutorials: http://wiki.ros.org/ROS/Tutorials
Linux basics: http://www.ee.surrey.ac.uk/Teaching/Unix/

There is a Troubleshooting page at the end of this document.
There is a document located on the desktop named Commands.txt which contains all the commands mentioned in this tutorial.

1. Plugin the System

ROS master runs on the RPI located in the locker, plug in the ethernet cable, Arduino and power supply (Optional to plugin HDMI, mouse and keyboard).

2. Connecting to the RPI

Two ways to SSH to the RPI:
SSH the RPI through the Host PC using putty, open putty and load the save session "MCSIM RPI" (Password: New4you).

SSH the RPI through Eduroam from any computer, this IP address is dynamic and at the time "130.229.176.24". username: mcsim password: New4you.

3. Starting ROS

The ROS system consists of 3 nodes, Simulink, Unity3D and the Arduino. Beyond that, you need to start the ROS master so all the nodes can connect to the master.

Starting the ROS Master

3 steps to start the master and its containing protocols. There is a Easy start-up and a manual start-up.

1. Starting the ROS core on the master
2. Starting the ROS bridge protocol which handles the communication with Unity
3. Starting the ROS serial protocol which handles the communication with the Arduino.

Easy start-up: write "roslaunch master system.launch" in the bash.

One command which will run a launch file, this presumes that everything is connected properly.
Command: **"roslaunch master system.launch",** this launch file is located in the master package. The master package is in the "catkin_MCSIM" folder.

Manual start-up:

Start the ROS core writing **"roscore"** in bash.
Start the ROS bridge writing **"roslaunch rosbridge_server rosbridge_websocket.launch"** in bash.
Start ROS serial writing **"rosrun rosserial_python serial_node.py /dev/ttyACM0"** (Make sure the right port is setup) in bash.

Starting ROS in Matlab/Simulink

Open MC model

ROS Workspace – Map structure

ROS works inside a workspace, where you have the packages and these packages contains the nodes that will be the subscriber/publisher. Furthermore, the packages include the custom messages needed for the nodes inside the package.

Unity Package

In Unity3D the nodes are configured locally using C#, to publish you extract what you want from the game model and the convert it in a way that it can be understood in the ROS network. The custom messages are also programmed in C#. The same goes for subscriptions, you subscribe to a topic in ROS and then you convert it in a way that it can be connected to an object in Unity3D

Matlab/Simulink Package

In Matlab/Simulink you can use the publish/subscribe blocks in Simulink. To make custom messages, you need to create the message in a ROS environment and then use a plugin to convert them to (Matlab/Simulink).

Arduino Package

The Arduino Environment is controlled from the IDE on the RPI. Rosserial is installed on the RPI ROS as a plugin and is used to communicate with the Arduino.

Troubleshooting

Here are some tips on what to look at when the communication system is not working.

USB port for the Arduino

Open the Arduino IDE on the RPI, serial port should be “/dev/ttyACM0”. If this is not the case, go to “~/catkin_MCSIM/src/master/launch”, edit the file “system.launch” using the command “sudo nano system.launch” like the figure below and change “args” to the right port.

```
mcsim@mcsim:~/catkin_MCSIM/src/master/launch$ sudo nano system.launch
```

IP address of the RPI

Typing the command “hostname -I” will give you the ip addresses.

```
mcsim@mcsim:~$ hostname -I
192.168.10.1 130.229.176.24 2001:6b0:1:1041:f549:e440:6da9:5c50
```

The first address is the ethernet connection to the Host PC “192.168.10.1”, this is static.
The second address is the wifi connection to Eduroam.