ABB IRB 1600 ROS Setup

Written By Andrew Schueler

Now that I have gotten the robot to move, I have compiled here the steps I used to get the environment installed on my machine. These were mainly quick notes and mild scratch work, but I think it gets the idea across. I should thank the MQP team for pointing me in the right direction while I created this list of steps. If anybody feels there are changes that need to be made, do mention it. I also should mention that, since some components are out of order, you should read the document in its entirety before embarking on the installation of ROS Industrial and MoveIt!.

First, you will need to make sure you have ros-kinetic-desktop-full installed on your ubuntu 1604 linux machine. My ROS catkin workspace is located here:

/home/$USER/catkin_ws/

This may be different on your system. If you have not yet done this installation, there are tutorials elsewhere that are way better than anything I could draft up.

I also have some environment variables set up in my ~/.bashrc file.

```
# This is the line that connects my path to ROS 1 setup.bash.
source /opt/ros/kinetic/setup.bash
# I have to source this too.
source ~/catkin_ws/devel/setup.bash
# This is so that roscore knows where to run
export ROS_MASTER_URI=http://192.168.0.23:11311
export ROS_HOSTNAME=192.168.0.23
```

Supposedly you do not need to set ROS_MASTER_URI or ROS_HOSTNAME, but I seem to get more stable results setting these environment variables.

To find out what to set ROS_HOSTNAME to, I ran

`ifconfig`

on the terminal and then looked at where the phrase

`inet addr: xxx.xxx.xxx.xxx`

showed up.
Then, for ROS_MASTER_URI, you run the same command on the machine you want to run roscore on, and then simply append `:11311` to the end and `http://` to the front.

Make sure to run

`source ~/.bashrc`

once that is all done.

Now you need to `cd` into your catkin workspace source folder, which is for me

`~/catkin_ws/src/`

You will need to git clone some repositories into your workspace.

Git clone this one for kinetic-devel
https://github.com/ros-industrial/abb_experimental

Run `git branch -a` to figure out what branch you are currenly on. If you are on the wrong branch, run

`git checkout kinetic-devel`

that will put you on the correct branch.

Next, you will want to go to the link provided here and run the commands

`sudo apt-get install xxxxxxx`

http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/trac_ik/trac_ik_tutorial.html

You will also have to apt install this thing:

sudo apt-get install ros-kinetic-tf2-geometry-msgs

Now you need to install MoveIt! Go to this link to do this:

http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/getting_started/getting_started.html

I expect that, when you get to the second page, you cannot go further. This is because you need to do the advanced setup for MoveIt since you still do not have all of the necessary packages set up. I am almost certain you will have this problem. As a result, you will have to build from source. I believe the MQP team said they had to do this. Here is the link to do this:

http://moveit.ros.org/install/source/

Make sure that you go to the bottom of the page and notice the phrase

rosdistro

This is Kinetic for this installation.

As a reminder, this is the link to the main website for the MoveIt libraries:

https://moveit.ros.org/

You now need the abb robot driver. This is in the non-experimental abb repository for ros industrial. This is the link you need to git clone.

https://github.com/ros-industrial/abb.git

You may not have to git clone `abb` because building from source automatically git clone the necessary packages to your catkin workspace.

NOW that you have downloaded all of those repositories into your workspace, you can finally run

`catkin_make`

in

`/home/$USER/catkin_ws/`

I should note that you should NOT catkin_make until everything is git cloned, installed, or built. Also, where the MoveIt advanced install instructions say to do `catkin build`, you really want to do catkin_make.

Once you have gotten the installations squared away, we will have to do some manual modifications of launch files before you connect the robot to ROS.

In this ros package:

`abb_experimental`

There is a subpackage:

`abb_irb1600_6_12_moveit_config`

With this launch file:

moveit_planning_execution.launch

You need to edit this launch file to decouple joints 2 and 3.

For me, this file is located here:

`/home/$USER/catkin_ws/src/abb_experimental/abb_irb1600_6_12_moveit_config/launch`

I do not remember what the old version of the file looks like, but I only commented out lines. I did not delete anything.

```
<launch>
<!-- The planning and execution components of MoveIt! configured to run -->
<!-- using the ROS-Industrial interface. -->
<!-- Non-standard joint names:
- Create a file [robot_moveit_config]/config/joint_names.yaml
controller_joint_names: [joint_1, joint_2, ... joint_N]
- Update with joint names for your robot (in order expected by rbt controller)
- and uncomment the following line: -->
<rosparam command="load" file="$(find abb_irb1600_support)/config/joint_names_irb1600_6_12.yaml" />
<!-- the "sim" argument controls whether we connect to a Simulated or Real robot -->
<!-- - if sim=false, a robot_ip argument is required -->
<arg name="sim" default="true" />
<arg name="robot_ip" unless="$(arg sim)" />

<!-- By default, we do not start a database (it can be large) -->
<arg name="db" default="false" />
<!-- Allow user to specify database location -->
<arg name="db_path" default="$(find abb_irb1600_6_12_moveit_config)/default_warehouse_mongo_db" />



<!-- Here we set the default linkage coupling. This needs to be false. -->
```

```
<arg name="link_2_3_coupled" default="false" doc="If true, compensate for J2-J3 parallel linkage" />



<!-- load the robot_description parameter before launching ROS-I nodes -->
<include file="$(find abb_irb1600_6_12_moveit_config)/launch/planning_context.launch" >
<arg name="load_robot_description" value="true" />
</include>

<!-- run the robot simulator and action interface nodes -->
<group if="$(arg sim)">
<include file="$(find industrial_robot_simulator)/launch/robot_interface_simulator.launch" />
</group>

<!-- run the "real robot" interface nodes -->
<!-- - this typically includes: robot_state, motion_interface, and joint_trajectory_action nodes -->
<!-- - replace these calls with appropriate robot-specific calls or launch files -->
<group unless="$(arg sim)">

<!-- I needed to change this. -->
  <!--
 <include file="$(find abb_irb1600_support)/launch/robot_interface_download_irb1600_6_12.launch" >
   <arg name="robot_ip" value="$(arg robot_ip)"/>
 </include>
  -->

<include file="$(find abb_irb1600_support)/launch/robot_interface_download_irb1600_6_12_modified.launch"
>
<arg name="robot_ip" value="$(arg robot_ip)"/>
<arg name="J23_coupled" value="$(arg link_2_3_coupled)"/> <!-- I am adding a param and setting it to false.
-->
</include>



</group>

<!-- publish the robot state (tf transforms) -->
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />

<include file="$(find abb_irb1600_6_12_moveit_config)/launch/move_group.launch">
<arg name="publish_monitored_planning_scene" value="true" />
</include>

<include file="$(find abb_irb1600_6_12_moveit_config)/launch/moveit_rviz.launch">
<arg name="config" value="true"/>
</include>
<!-- If database loading was enabled, start mongodb as well -->
<include file="$(find abb_irb1600_6_12_moveit_config)/launch/default_warehouse_db.launch" if="$(arg db)">
<arg name="moveit_warehouse_database_path" value="$(arg db_path)"/>
</include>

</launch>
```

This launch file described above also calls another launch file that also needs editing. The file you need to change is

robot_interface_download_irb1600_6_12.launch

I created my own version called:

robot_interface_download_irb1600_6_12_modified.launch

So that I do not have to call the original launch file, which is located here:

/home/$USER/catkin_ws/src/abb_experimental/abb_irb1600_support/launch

This modified launch file looks like this:

```
<!--
Manipulator specific version of abb_driver's 'robot_interface.launch'.

Defaults provided for IRB 1600:
- J23_coupled = true

Usage:
robot_interface_download_irb1600.launch robot_ip:=<value>
-->
<launch>
<arg name="robot_ip" doc="IP of the controller" />
<!-- Andrew moved this line into a comment.
<arg name="J23_coupled" default="true" doc="If true, compensate for J2-J3 parallel linkage" />
-->

<arg name="J23_coupled" default="false" doc="If true, compensate for J2-J3 parallel linkage" />

<rosparam command="load" file="$(find abb_irb1600_support)/config/joint_names_irb1600_6_12.yaml" />

<include file="$(find abb_driver)/launch/robot_interface.launch">
<arg name="robot_ip" value="$(arg robot_ip)" />
<arg name="J23_coupled" value="$(arg J23_coupled)" />
</include>
</launch>
```

Overall, modified the launch files above. I turned the couple_J_2_3 off as well as set the default IP address. Other than that, everything else is left alone.

I did make one other change to the launch files. This one should NOT be necessary, nor is it recommended, but I have included it here so that I am not missing anything as I happened to make these changes in the process. I just have not gotten to changing them back to the original version on my ubuntu

laptop. I edited this .xml file:

`trajectory_execution.launch.xml`

(It is in the same directory as the upper level launch file, If I am not mistaken.)

And I got the idea to edit it from this post:

https://answers.ros.org/question/196586/how-do-i-disable-execution_duration_monitoring/

Here is what the file looks like for me:

```
<launch>

<!-- This file makes it easy to include the settings for trajectory execution -->

<!-- Flag indicating whether MoveIt! is allowed to load/unload or switch controllers -->
<arg name="moveit_manage_controllers" default="true"/>
<param name="moveit_manage_controllers" value="$(arg moveit_manage_controllers)"/>

<!-- Andrew added this line from here:
https://answers.ros.org/question/196586/how-do-i-disable-execution_duration_monitoring/

<param name="trajectory_execution/execution_duration_monitoring" value="true" />
-->

<!-- When determining the expected duration of a trajectory, this multiplicative factor is applied to get the
allowed duration of execution -->
<param name="trajectory_execution/allowed_execution_duration_scaling" value="1.2"/> <!-- default 1.2 -->
<!-- Allow more than the expected execution time before triggering a trajectory cancel (applied after scaling) --
>
<!-- ANdrew played with this too.-->
<param name="trajectory_execution/allowed_goal_duration_margin" value="3.5"/> <!-- default 0.5 -->

<!-- Allowed joint-value tolerance for validation that trajectory's first point matches current robot state -->
<!-- Andrew set the default from 0.01 to 0.0-->
<param name="trajectory_execution/allowed_start_tolerance" value="0.1"/> <!-- default 0.01 -->
<!-- Load the robot specific controller manager; this sets the moveit_controller_manager ROS parameter -->
<arg name="moveit_controller_manager" default="abb_irb1600_6_12" />
<include file="$(find abb_irb1600_6_12_moveit_config)/launch/$(arg
moveit_controller_manager)_moveit_controller_manager.launch.xml" />
</launch>
```

Now that I have the ROS Node working, we have to now look at the robot arm itself and the teach pendant.

The first order of business is to boot the robot arm into

`System2_ROS`

You need to select

`Control Panel`

and then look down low at the last option in the list above the boot options. If I remember correctly, it contains buzzwords like

“activation” and “select system”

or something like that. I am just forgetting at the moment.

You then need to select the System2_ROS option and then look down and to the right and select

`Activate`

and then accept whatever the prompt complains about. The pendant will then reboot to the ROS arm configuration. Once that is done, you need to click on the “ABB” in the upper left and then select

`Program editor`

option. Then you select “Debug” on the bottom and then select “PP to main” and then you should be set to go…

NOPE! NEVER FORGET THIS LAST STEP!

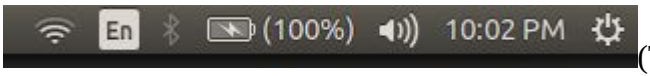
Press play on the teach pendant. Only then can you execute trajectories.

ONE MORE STEP BEFORE ROS WILL WORK

When setting up networking for ROS, you can set the inet addr to the wifi address the WPI network gives you. HOWEVER, you need to set a STATIC IP ADDRESS, or the robot arm won’t connect to your machine.

You can do this via the Ubuntu GUI interface since this is the easiest method for most people.

Left click on the wifi icon in the upper right hand corner:



(The button farthest to the left)

Then select “Edit Connections”.

Then, under the Ethernet dropdown, select the only wired connection available, and click “EDIT”. If you do not have one, click on the text “Ethernet” and click “ADD”.

Move over to the “IPv4” tab and set “Method” to “Manual”.

Select “ADD” to the right and then set the “Address” to any ip that is not currently in use. For me, I use “192.168.100.123”. Then, set the “Net mask” to “24” and leave the “Gateway” blank. Then, click “Save” in the bottom right. Now the static ip address of the ethernet port on your laptop has been configured.

Now that you have installed everything, modified those 2/3 xml/launch files, you can now run this command to start the simulated robot up with ROS:

`roslaunch abb_irb1600_6_12_moveit_config moveit_planning_execution.launch robot_ip:=192.168.100.100 sim:=true`

Or the real robot with ROS:

`roslaunch abb_irb1600_6_12_moveit_config moveit_planning_execution.launch robot_ip:=192.168.100.100 sim:=false`

Notice that this is actually one long line in the terminal. It just looks weird right now.

Just some final notes:

I find that the ROS stuff is more stable if you start the robot “pp to main” first, *then* run the launch file. You also do not need to start roscore separately as the launch file will do that for you. If you run into issues with roscore not working, you probably need to make sure you set the environment variables properly.

I should also mention that the static ip configuration should be done **BEFORE** running the robot arm RAPID program. I did not really explain that well earlier. This is a rough draft...ss

I believe that this is everything required to get the arm to work. If somebody decides to do this, they should call the uppermost launch file described earlier in their own custom launch file in their own custom ROS package so that they do not have any of the ROS-Industrial packages in their personal git repositories.