


BAXTER USER'S GUIDE



TABLE OF CONTENTS

Contents

List of Figures	3
INTRODUCTION	4
Prerequisites	4
Videos	4
Research Baxter Videos For Examples	4
Baxter Research Robot Example: Move Baxter’s Arms Using Keyboard	4
Baxter Research Robot Examples: Puppet.....	4
Simulator Videos	4
Manufacturing Baxter Videos	5
Customer Videos- Many Examples of Baxter Applications	5
Human-Robot Interaction	5
Machine Learning	5
Planning and Manipulation	5
Manipulation and Mechatronics	5
Computer Vision	5
Baxter Research Robot Speaks Out	6
• https://www.youtube.com/watch?v=LOn5WoTnkQU	6
LOG IN and ENABLE BAXTER	7
 ENABLE BAXTER	7
Before Baxter respond to any commands, the robot must be enabled. The WEB page describes the Enable_Robot_Tool and the video shows how Baxter is enabled.	7
http://sdk.rethinkrobotics.com/wiki/Enable_Robot_Tool	7
Functions- Options	7
VERIFY CONNECTION: Echo Baxter's joint_states	8
To Disable the robot: Note – Tuck the arms first.....	8
SDK EXAMPLES (wiki/FOUNDATIONS)	10
Fundamental Examples	10
Other Baxter Rethink Examples that are not discussed in this section are listed in Appendix I.	10
RUN EXAMPLE PROGRAMS	10
Example 1 Run an example program to wobble arms:	10
Example 2. Joint Position Keyboard Example	11

Example 3. Joint Position Waypoints Example	13
Example 4. Joint Trajectory Playback Example	15
Example 5. Put a picture on Baxter’s face	18
Example 6. Baxter’s Cameras	19
Camera Control Example	20
Example 7. This example will blink the LED on the Left Navigator on and then off.	22
Carol’s Easy Script for Baxter Commands	23
SIMULATORS	24
Turtlesim and Introduction To ROS	25
MoveIt	35
Gazebo	39
APPENDIX I BAXTER RETHINK EXAMPLES	42
Movement	42
Input and Output	42
Examples with Links From FOUNDATIONS 8/30/2014 http://sdk.rethinkrobotics.com/wiki/Examples	44
APPENDIX II Summary of Ubuntu and ROS commands	45
APPENDIX III ROS Workspace Directories	46
APPENDIX IV Carol’s Script for .run_baxter	47

List of Figures

Figure 1 Baxter's Cuff for Zero-G mode.....	8
Figure 2 Baxter's Arms Untucked and Tucked	9
Figure 3 Baxter's Arm and Joint Designations.....	11
Figure 4 Baxterworking.png	18
Figure 5View from Baxter's Head Camera	20
Figure 6 Turtlesim 1 with the Turtle after the node is executed	26
Figure 7 Turtlesim After Moving.....	27
Figure 8 Four Turtlesim Windows using Terminator.....	29
Figure 9 Turtle responds to published topic.....	32
Figure 10 Turtlesim graph showing communication	34
Figure 11MoveIt and BaxterInitial View	36
Figure 12 Simulator with Initial and Desired Position of Arms	37
Figure 13Baxter Simulator Ready for Execution of Moves	37
Figure 14Baxter Simulation Showing Final Position of Arms	38
Figure 15 Gazebo Simulation of Baxter before Enabling Baxter	40
Figure 16 Baxter Enabled in Simulation.....	40
Figure 17 Baxter Software	43

INTRODUCTION

This report describes the Rethink Robotics examples for Baxter the robot. The instructions to enable Baxter and run the examples are presented. The report also describes several types of simulation of Baxter as well as the Turtlesim simulator useful for learning ROS.

Prerequisites

Before using Baxter in the UHCL lab, you should have read and understood the material covered in the *Introduction to Baxter* report by T.L. Harman and Carol Fairchild. In the lab, Baxter has been setup and can be commanded by Terminal Commands that execute scripts. You should have an account on the workstation and be able to login.

Videos

As a useful prerequisite to using Baxter, it would be helpful to view a number of videos that show the various examples of Baxter in action.

Research Baxter Videos For Examples

Meet the Baxter Research Robot (General)

http://www.youtube.com/watch?feature=player_embedded&v=G2-4WFr9-X0

Baxter Research Robot Example: Enable Robot

<https://www.youtube.com/watch?v=tYpNk5v7wfl>

Baxter Research Robot Example: Move Baxter's Arms Using Keyboard

[Baxter Research Robot Examples: Joint Position Using Keyboard](#)

Baxter Research Robot Examples: Puppet

<https://www.youtube.com/watch?v=TTgwcczfCJQ>

Baxter Research Robot Record-Playback Example

https://www.youtube.com/watch?v=uk1XBMsNhco&feature=player_detailpage

Simulator Videos

MoveIT Video

https://www.youtube.com/watch?feature=player_embedded&v=1Zdkwym42P4

Manufacturing Baxter Videos

Baxter Folds a Shirt

https://www.youtube.com/watch?v=Mr7U9pQtwq8&feature=player_detailpage

Customer Videos- Many Examples of Baxter Applications

http://sdk.rethinkrobotics.com/wiki/Customer_Videos

Human-Robot Interaction

- [David Using Jammster](#)
- [Magic Robot - The Illusion of the Thinking Machine](#)
- [Baxter on wheels retrieving jacket](#)
- [Baxter Robot control using body tracking with kinect](#)
- [Clothing and Unclothing Assistance by Baxter](#)
- [Using a Baxter Robot for Co-Operative Disassembly of a CPU](#)
- [Teleoperation of Baxter Robot using Phantom Omni](#)

Machine Learning

- [Human touch makes robots smarter: On Learning Context-Driven User Preferences](#)
- [Baxter Experiments for Deep Learning for Detecting Robotic Grasps](#)

Planning and Manipulation

- [Robot Motion Planning for Reactive Execution of Learned Tasks](#)
- [Baxter Coordinated Dual-Arm Force Control](#)
- [Baxter Research Robot Solves Rubik's Cube](#)
- [Human touch makes robots smarter: On Learning Context-Driven User Preferences](#)
- [Baxter Research Robot: Mimicry using Kinect](#)
- [Online human upper body imitation using BAXTER robot](#)

Manipulation and Mechatronics

- [Baxter Research Robot at WPI with Prof. Dmitry Berenson](#)
- [Optimal Parameter Identification of Flexible Objects via Manipulation](#)
- [Teleoperating Multiple Baxter Robots Using Kinect v2](#)
- [Soft Pneumatic Robot Hand](#)
- [Baxter Robot Pipetting Complete](#)
- [Packaging Demo of Baxter Robot using 2-Finger Adaptive Robot Grippers from Robotiq](#)
- [GelSight sensor gives robots touch](#)

Computer Vision

- [Happy Easter from the RRC Robotics and Automation Team](#)
- [Automated Lego Sorting](#)
- [Bartender Baxter](#)
- [Towards an Automated Checked Baggage Inspection System Augmented with Robots](#)
- [Handle Localization in 3D Point Clouds](#)
- [BAXTER Dunks A Ball](#)
- [BAXTER Sort Colored Balls - Author's View](#)

- [Baxter Perfect Colored Cube Sort](#)

Baxter Research Robot Speaks Out

- <https://www.youtube.com/watch?v=LOn5WoTnkQU>

LOG IN and ENABLE BAXTER

Before running examples on Baxter, it is necessary to login on the workstation using your password. Then, go to the ROS workspace and execute the baxter shell which sets up communication between the workstation and Baxter. Our commands will be in **bold text** after the \$ command prompt in a terminal window. The response will be in DejaVu Sans Mono font for baxter.

1. Log in (1. Log in using your password)
2. Begin a Terminal Session (2. Create a terminal window and change directories as in 3.)

```
$ cd /home/tlharmanphd/ros_ws (3. Go to the ROS workspace and execute Baxter Shell)  
$ ./baxter.sh (4. Enable communication with Baxter)
```

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$
```

```
$ roslaunch baxter_tools tuck_arms.py -u (5. Enable Baxter and Untuck his arms)
```

The last command is in the typical ROS format:

```
roslaunch <package name> <ROS node name> <options>
```

Now the command line shows baxter and the IP address of the robot. This is one purpose of the baxter shell baxter.sh.



ENABLE BAXTER

Before Baxter respond to any commands, the robot must be enabled. The WEB page describes the Enable_Robot_Tool and the video shows how Baxter is enabled.

http://sdk.rethinkrobotics.com/wiki/Enable_Robot_Tool

https://www.youtube.com/watch?feature=player_embedded&v=tYpNk5v7wfl&x-yt-ts=1422579428&x-yt-cl=85114404

This tool is responsible for enabling (powering and state monitoring) Baxter. Enabling the robot is expected and necessary for standard Baxter usage.

A fundamental tool for use when working with Baxter, the enable_robot tool, provided in the baxter_tools SDK package, allows for enabling/disabling/resetting/stopping the robot. Baxter must be enabled in order to actively command any of the motors.

Baxter will now be enabled. The joints will be powered, and Baxter will hold his current joint positions with a position control loop.

Functions- Options

To get help on enable_robot use the -h argument:


```
$ rosrn baxter_tools enable_robot.py -h
```

Help screen:

```
enable_robot.py [ARGUMENTS]
```

- ⑩ h, --help show this help message and edit
- ⑩ s, --state Print current robot state
- ⑩ e, --enable Enable the robot
- ⑩ d, --disable Disable the robot
- r, --reset Reset the robot
- ⑩ S, --stop Stop the robot

VERIFY CONNECTION: Echo Baxter's joint_states

```
$ rostopic echo /robot/joint_states (LOTS OF STATES)
```

You should see a continuous stream of Baxter's joint names with measured positions, velocities and torques.

All is well! You have successfully setup communication between your development PC and Baxter if the joint states are displayed in the terminal window.

Move Baxter's arms by grabbing Baxter's cuff:

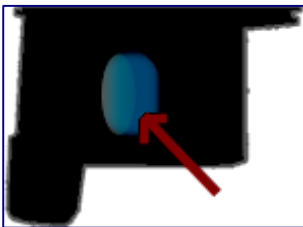


Figure 1 Baxter's Cuff for Zero-G mode

Once Baxter is enabled, Baxter's arms are in the "Zero-G" mode. The position control loop will be released with solely gravity compensation enabled. This allows for intuitive hand-over-hand guidance of the limbs throughout the workspace. Baxter's arms will move according to the motion you impart to them. This method is used to train Baxter to reach various positions and is used in the Joint Record and Playback example described later.

To Disable the robot: Note – Tuck the arms first

```
$ rosrn baxter_tools tuck_arms.py -t  
$ rosrn baxter_tools enable_robot.py -d
```

```
TO LEAVE BAXTER CNTL+D
```

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ cntl+d  
tlharmanphd@D125-43873:~/ros_ws$
```

TUCK AND UNTUCK ARMS http://sdk.rethinkrobotics.com/wiki/Tuck_Arms_Tool

```
$ rosrun baxter_tools tuck_arms.py -u
```

```
$ rosrun baxter_tools tuck_arms.py -t
```

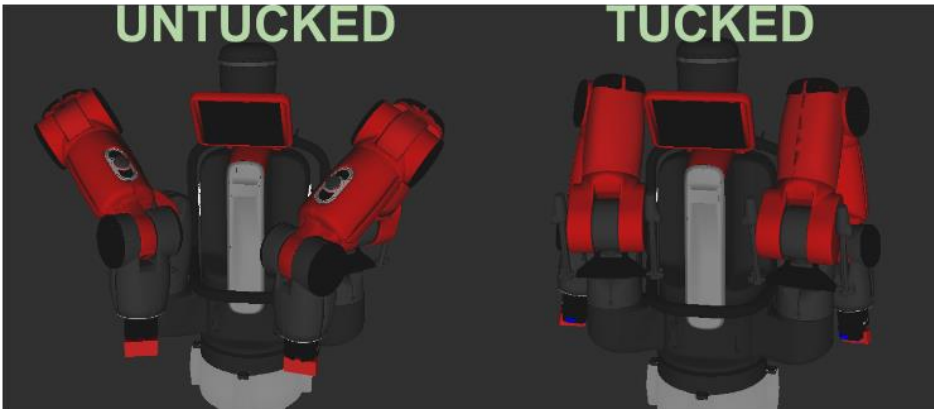


Figure 2 Baxter's Arms Untucked and Tucked

SDK EXAMPLES (wiki/FOUNDATIONS)

SDK Examples

The SDK Example Programs from Rethink Robotics are designed to demonstrate using the various interfaces and features of Baxter. By following the Usage Guide on an Example Page on the wiki Website, you can try out some of Baxter's functionality. Each example also has a corresponding Code Walkthrough that will take you through the program and explain how Rethink Robotics uses the interfaces. The code walkthroughs are more advanced and are not covered in this report.

Fundamental Examples

Enable Robot Example - This tool is responsible for enabling (powering and state monitoring) Baxter. Enabling the robot is expected and necessary for standard Baxter usage.

http://sdk.rethinkrobotics.com/wiki/Enable_Robot_Tool

Other Baxter Rethink Examples that are not discussed in this section are listed in Appendix I.

RUN EXAMPLE PROGRAMS

A number of Baxter example programs are provided which use the `baxter_interface` package which contains Python modules for Baxter Research Robot development.



Be sure that the script `baxter.sh` has been executed in the terminal window and Baxter is Enabled.

Example 1 Run an example program to wobble arms:

[Wobbler Example](#) Use wobbler as an example of controlling Baxter using joint velocity control. Arms “wobble”

```
$ rosrn baxter_examples joint_velocity_wobbler.py
```

This example will simply move the arms to a neutral position, enter into velocity control mode, moving each joint through a random sinusoidal motion. More about this example on the Joint Velocity Wobbler Example Page. **Press Ctrl-C to stop...**

Example 2. Joint Position Keyboard Example

This example demonstrates control of Baxter's joints using the keyboard to move the joints. It is a good example to show how Baxter's arms move in response to joint commands. For example, try to move Baxter's arm with the keys to pick up an object as a simple project. It is not as easy as you might think at least not the first couple of times!

VIEW the WEB page: [Joint Position Example](#)

The Joint Position Control Examples demonstrate how to use position control to move and sense the arm based on joint angles. Examples include keyboard or joystick based user control of arm angles, along with example programs from Rethink to record and playback joint positions.

VIEW VIDEO

[Baxter Research Robot Examples: Joint Position Using Keyboard](#)

1. Power on Baxter - white button on rear
2. LOG ON Workstation
3. Go to `~/ros_ws` (ROS workspace)
4. `./baxter.sh`
5. `roslaunch baxter_tools enable_robot.py -e`
6. `roslaunch baxter_examples joint_position_keyboard.py`

TYPE ? FOR LIST OF COMMAND KEYS TO MOVE JOINTS WITH NAMES SHOWN IN FIGURE:

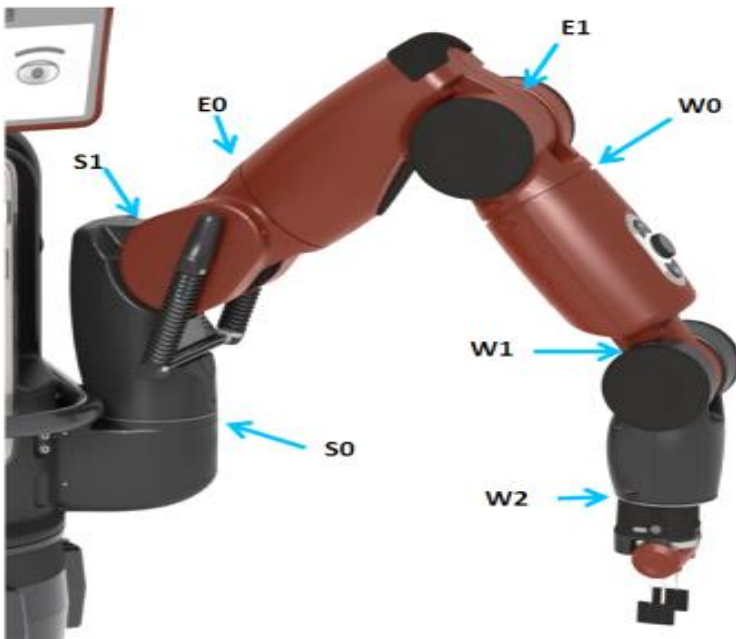


Figure 3 Baxter's Arm and Joint Designations

EXAMPLE AT TERMINAL WINDOW

```
tlharmanphd@D125-43873:~$ cd ~/ros_ws
tlharmanphd@D125-43873:~/ros_ws$ ./baxter.sh
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ rosrun baxter_tools
enable_robot.py -e
```

```
[INFO] [WallTime: 1422042340.373074] Robot Enabled
```

```
[baxter - http://172.29.64.200:113
```

```
1] tlharmanphd@D125-43873:~/ros_ws$ rosrun baxter_examples joint_position_keyboard.py
```

```
  Initializing node...
```

```
  Getting robot state...
```

```
  Enabling robot...
```

```
[INFO] [WallTime: 1422042352.890294] Robot Enabled
```

```
[WARN] [WallTime: 1422042353.064890] left_gripper electric: Gripper Firmware version (3.0.0
5.5) does not match SDK Version (1.0.0). Use the Robot's Field-Service-Menu to Upgrade your
Gripper Firmware.
```

```
Controlling joints. Press ? for help, Esc to quit.
```

```
key bindings:
```

Left Arm and Gripper	Right Arm and Gripper
?: Help	RIGHT
/: left: gripper calibrate	b: right: gripper calibrate
,: left: gripper close	c: right: gripper close
m: left: gripper open	x: right: gripper open
y: left_e0 decrease	q: right_e0 decrease
o: left_e0 increase	r: right_e0 increase
u: left_e1 decrease	w: right_e1 decrease
i: left_e1 increase	e: right_e1 increase
6: left_s0 decrease	1: right_s0 decrease
9: left_s0 increase	4: right_s0 increase
7: left_s1 decrease	2: right_s1 decrease
8: left_s1 increase	3: right_s1 increase
h: left_w0 decrease	a: right_w0 decrease
l: left_w0 increase	f: right_w0 increase
j: left_w1 decrease	s: right_w1 decrease
k: left_w1 increase	d: right_w1 increase
n: left_w2 decrease	z: right_w2 decrease
.: left_w2 increase	v: right_w2 increase
Esc: Quit	

Example 3. Joint Position Waypoints Example

This is a basic example for joint position moves. Move Baxter's arms using the zero_G mode and record a number of joint position waypoints using the round navigation button. These waypoints will then be played back upon completion of the moves.

Description

The joint position waypoints example demonstrates basic joint position control. A joint position waypoint is a configuration of the arm in joint space (ie. simultaneous joint angles for all of the arm's seven degrees of freedom). In this example, the robot is enabled to move the specified limb in zero-g mode. Using the arm's navigator buttons, a sequence of joint position waypoints can be recorded. Upon completion of recording, the limb loop will be commanded through the recorded joint sequence.

Usage

Verify that the robot is enabled.

Start the joint position waypoints example program, specifying the `left` or `right` arm:

```
$ rosrun baxter_examples joint_postion_waypoints.py -l right
```

A prompt will provide instructions for recording joint position waypoints:

```
Initializing node...
Getting robot state...
Enabling robot...
[INFO] [WallTime: 1412620077.078718] Robot Enabled
[INFO] [WallTime: 1412620077.174666] Waypoint Recording Started
Press Navigator 'OK/Wheel' button to record a new joint joint position waypoint.
Press Navigator 'Rethink' button when finished recording waypoints to begin playback
```

MOVE THE ARM IN ZERO-G MODE. When at a joint position configuration you would like to record, PRESS THAT LIMB'S NAVIGATOR 'WHEEL'. Feedback will be displayed that the joint position waypoint has been recorded:

```
Waypoint Recorded
...
```

WHEN DONE RECORDING WAYPOINTS, PRESS the limb Navigator's 'Rethink' button and playback will begin:

```
[INFO] [WallTime: 1399571721.540300] Waypoint Playback Started
Press Ctrl-C to stop...
```

The program will begin looping through the recorded joint positions. When a joint position waypoint is fully achieved (within the accuracy threshold), the next recorded joint position will be commanded.

```
Waypoint playback loop #1
Waypoint playback loop #2
...
```

Pressing Control-C at any time will stop playback and exit the program.

The parameter and options for the joint position waypoints example are:

Required argument:

-l or --limb The limb (arm) on which the waypoints will be captured.

Optional arguments:

-h or --help Help

-s or --speed The joint position motion speed ratio [0.0-1.0] (default = 0.3)

-a or --accuracy The threshold in Radians at which the current joint position command is considered successful before sending the next following joint position command. (default = 0.008726646)




This ridiculously accurate looking value represents

$0.008726646 * 180/\pi = 0.5$ degrees

Example 4. Joint Trajectory Playback Example

The example shows how to Record and Playback Arm and Gripper Positions. Define a recorder file and then move the arms while holding the cuffs. The joint positions with corresponding timestamps will be recorded for both arms. NOTE: You can open and close the grippers while recording by using Baxter's cuff buttons:

Oval = Close, Circle = Open Press any key to exit when done recording. The movements can be played back one or more times according to the option set for playback

 This example uses **two** terminal windows. One is for recording and one is for playback since the `joint_trajectory_action_server.py` script must be running in one window for the playback to work. In each window, make sure that the directory is the ROS workspace (`ros_ws`) and the `baxter.sh` shell is executed to allow communication with Baxter.

The WEB page describes the application and the YouTube video shows how to record and playback the positions:

http://sdk.rethinkrobotics.com/wiki/Joint_Trajectory_Playback_Example

https://www.youtube.com/watch?v=uk1XBMsNhco&x-yt-ts=1422579428&x-yt-cl=85114404&feature=player_embedded

We have defined the file “`my_first_recording.rec`” to hold the time and joint and gripper positions for this example. The `joint_recorder` script is in the `baxter_examples` package.

1. Execute this command with your file <recordingfile> (Note the `-f` option):

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ rosrunc baxter_examples
joint_recorder.py -f my_first_recording.rec
```

```
Initializing node...
```

```
Getting robot state...
```

```
Enabling robot...
```

```
[INFO] [WallTime: 1422556692.423999] Robot Enabled
```

```
Recording. Press Ctrl-C to stop.
```

2. `^C` (Cntl-c after moving arms and manipulating the gripper to stop recording)

```
Done.
```

3. Check to see the recording file was created

```
$ ls (To view directory and your new file)
```

```
-rw-rw-r-- 1 tlharmanphd tlharmanphd 2005277 Jan 29 12:39 my_first_recording.rec
```

(If viewed using a text editor you can view the positions of the joints and the gripper state.)

4. Execute the Playback Help command:

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ roslaunch baxter_examples joint_trajectory_file_playback.py -h
```

```
usage: joint_trajectory_file_playback.py [-h] -f PATH [-l LOOPS]
```

```
RSDK Joint Trajectory Example: File Playback
```

```
Plays back joint positions honoring timestamps recorded via the joint_recorder example.
```

```
Run the joint_recorder.py example first to create a recording file for use with this example. Then make sure to start the joint_trajectory_action_server before running this example.
```

```
This example will use the joint trajectory action server with velocity control to follow the positions and times of the recorded motion, accurately replicating movement speed necessary to hit each trajectory point on time.
```

```
optional arguments:
```

```
-h, --help          show this help message and exit  
-f PATH, --file PATH path to input file  
-l LOOPS, --loops LOOPS  
                    number of playback loops. 0=infinite.
```

```
Related examples:
```

```
joint_recorder.py; joint_position_file_playback.py.
```

5. Start the joint_trajectory_action_server

A commonly used ROS method for robot arm motion control is the joint trajectory action interface. The trajectory_controller and its corresponding joint trajectory action server is the baxter_interface implementation to support this action interface.

[http://sdk.rethinkrobotics.com/wiki/Simple Joint trajectory example](http://sdk.rethinkrobotics.com/wiki/Simple_Joint_trajectory_example)

Execute the command:

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ roslaunch baxter_interface joint_trajectory_action_server.py
```

```
Initializing node...
```

```
Initializing joint trajectory action server...
```

```
Running. Ctrl-c to quit
```

6. Playback the movements and gripper states

2nd Terminal Click on Terminal Icon, Open A New Window, and communicate with Baxter

```
tlharmanphd@D125-43873:/$ cd /home/tlharmanphd/ros_ws
```

```
tlharmanphd@D125-43873:~/ros_ws$ ./baxter.sh (Always execute baxter.sh in a new window)
```

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$
```

Start the playback:

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ rosrund baxter_examples  
joint_trajectory_file_playback.py -f my_first_recording.rec
```

```
Initializing node...
```

```
Getting robot state...
```

```
Enabling robot...
```

```
[INFO] [WallTime: 1422558116.740928] Robot Enabled
```

```
Running. Ctrl-c to quit
```

```
Playback loop 1 of 1
```

```
Exiting - File Playback Complete
```

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$
```

Here is an example of the recorded data starting with time and then listing the states of the joints and grippers. See Figure 3 for a definition of Baxter's joint and arm designations:

```
time, left_s0, left_s1, left_e0, left_e1, left_w0, left_w1, left_w2, left_gripper, right_s0, right_s1, right_e0, right_e1, right_w0, right_w1, right_w2, right_gripper
```

```
3.034468, -0.081300981665, -0.993252559021, -
```

```
1.18653413807, 1.94278666564, 0.667665137164, 1.02508266033, -0.498543755493, 100.0, 0.0839854480408, -
```

```
1.00245644374, 1.18269918611, 1.94201967524, -0.667665137164, 1.0285341171, 0.498543755493, 100.0
```

(Gripper Open)

Left Gripper Closing at about 15 seconds

```
15.005002, -0.0648106882141, 0.266529161591, -1.33379629354, 1.46188368918, 0.885873903992, -
```

```
0.385796167712, 1.33571376953, 99.5192337036 left_gripper (OPEN)
```

Closed

```
15.734782, -0.0674951545898, 0.275349551111, -1.33379629354, 1.46341766997, 0.885490408795, -
```

```
0.385796167712, 1.33571376953, 2.8846154213, left_gripper (CLOSED)
```

Opening

Example 5. Put a picture on Baxter's face

```
$ rosrn baxter_examples xdisplay_image.py --f baxterworking.png (Image in ros_ws)
```



WARNING: baxterworking.png 871.6 Kb (Digital Camera jpegs are too LARGE)

Screen Display – Example tool for displaying image files (png, jpeg) on the Head Screen.

Your computer, and the example will read and convert the image using cv_bridge, sending it to the screen as a standard ROS Image Message.

optional arguments:

-h, --help Show this help message and exit
-d SEC, --delay SEC Time in seconds to wait before publishing image

required arguments:

-f PATH, --file PATH Path to image file to send

Notes:

Max screen resolution is 1024x600.

Images are always aligned to the top-left corner.

Image formats are those supported by OpenCv – loadImage().

Replace the --file argument with the path to your own image.



Figure 4 Baxterworking.png

Example 6. Baxter's Cameras

http://sdk.rethinkrobotics.com/wiki/Camera_Control_Example

According to the WEB page: "There are three cameras available local to Baxter. A single camera is located in either of Baxter's hands, the 'left_hand_camera' and the 'right_hand_camera'. A third camera is available on Baxter's head, described as 'head_camera'. This example shows usage for listing available cameras, opening each of the three cameras with various parameters, and closing the cameras."



WARNING: Important Note: Due to limited bandwidth capabilities **only two** cameras can be operating simultaneously. Starting a camera while both of the other cameras are in operation will result in an error, and the camera will not open.

Important Note: Default behavior on Baxter startup is for both of the hand cameras to be in operation at a resolution of 320x200 at a framerate of 25 fps.

Supported Frame Size Modes: Frame sizes at which the cameras will operate.

- 1280x800
- 960x600
- 640x400
- 480x300
- 384x240
- 320x200

If given an unsupported frame size, this example will exit with the ValueError: Invalid Camera mode.

For more information on Baxter's cameras, see [Using the Cameras](#).

Usage

See the camera controller's usage on the command line by passing `camera_controller.py` the `-h`, help argument:

```
$ rosrn baxter_tools camera_control.py -h
```

Usage: `camera_control.py [-h] [-o CAMERA] [-c CAMERA] [-r RESOLUTION XxY] [-l]`

Optional Arguments

<code>-h, --help</code>	This screen
<code>-o, --open [CAMERA]</code>	Open specified camera
<code>-c, --close [CAMERA]</code>	Close specified camera
<code>-r, --resolution [X]x[Y]</code>	Set camera resolution
<code>-l, --list</code>	List available cameras

Camera Control Example

Verify that all cameras are closed using the `-c` option from a terminal session,

```
$ rosrun baxter_tools camera_control.py -c left_hand_camera
$ rosrun baxter_tools camera_control.py -c right_hand_camera
$ rosrun baxter_tools camera_control.py -c head_camera
```

The reason that we are closing all of the cameras is because of the note above describing the constraint that only two cameras can be operated simultaneously.

You can Open, Display and Close each of the three available cameras [left_hand_camera, right_hand_camera, head_camera] with various settings.

To list all of the available cameras:

```
$ rosrun baxter_tools camera_control.py -l
  (Get List in Terminal Window)

$ rosrun baxter_tools camera_control.py -o head_camera

$ rosrun image_view image_view image:=/cameras/head_camera/image
```



*Figure 5*View from Baxter's Head Camera

You can also View the camera feed in rviz. See the Camera Control Tool WEB page for more information:

http://sdk.rethinkrobotics.com/wiki/Camera_Control_Example

```
$ rosrun rviz rviz
```

Under the displays tab on the left hand side of rviz, change the 'Global Option - Fixed Frame' from '/map' to '/base'.

Select 'Add' in the displays tab of rviz.

Select 'Camera' display topic.

The 'Camera' topic will now be displayed in a new embedded window.

Under the 'Camera' tab on the left display window, choose the 'Image Topic':
`/cameras/right_hand_camera/image`

You should now see the `right_hand_camera` image in the embedded camera window. Other data displayed in rviz will also be projected onto this image accordingly (ie. turning on the 'RobotModel' display, and moving the right arm to point at the left arm will display the 'RobotModel' overlaid on your live video feed).

Example 7. This example will blink the LED on the Left Navigator on and then off.

\$ rosrun baxter_examples digital_io_blink.py \$ rosrun baxter_examples digital_io_blink.py -h
(Get Help)

RSDK Digital IO Example: Blink

(usage: digital_io_blink.py [-h] [-c COMPONENT_ID])

Turns the output of a DigitalIO component on then off again while printing the state at each step. Simple demonstration of using the `baxter_interface.DigitalIO` class. Run this example with default arguments and watch the light on the left arm Navigator blink on and off while the console echos the state. Use the `component_id` argument or ROS Parameter to change the DigitalIO component used.

Initial state: False New state: True Final state: False

optional arguments:

-h, --help show this help message and exit

-c COMPONENT_ID, --component COMPONENT_ID

name of Digital IO component to use (default: left_itb_light_outer)

Carol's Easy Script for Baxter Commands

Carol Fairchild has written a script that runs under Ubuntu with the name `. run_baxter`

(Note the `.` and space before `run_baxter`)

Here is the Help file for the script (shown in Appendix IV):

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ . run_baxter h
```

```
Today is Fri Jan 23 15:27:32 GST 2015
```

```
run_baxter commands:
```

```
enable, disable, state, reset, stop
```

```
tuck, untuck
```

```
arms_keyboard, record <filename>, playback <filename>
```

```
springs <right or left>, arms_wobbler, puppet <right or left>
```

```
ik <right or left>, joint_trajectory <right or left>
```

```
camera open <right, left or head> res <wide, medium or narrow>
```

```
camera close <right, left or head>
```

```
head_wobbler, gripper_keyboard, head_display <filename>
```

```
digital_io, analog_io
```

For example to execute the Enable command:

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ . run_baxter enable
```


SIMULATORS

This section covers several of the simulators that work with ROS. The first is called **Turtlesim** and is useful to learn about the ROS nodes, topics, and services. Turtlesim is a ROS **package** that contains executable code and other information to create a simple simulation using a moving “turtle”.

The second simulator considered here is called **MoveIt** and is used to simulate trajectory planning for Baxter the Robot as well as other popular robots. The trajectory can be sent to the physical Baxter and the simulated movements will be replayed.

The third simulator discussed is called **Gazebo** and like MoveIt can simulate Baxter and other robots.

Our introduction here is very brief to allow you to just “sample” the capability of these simulators. The use of MoveIt and Gazebo, for example, go far beyond simulating Baxter’s arm movements. They can be used for scene creation including planning trajectories with obstacle avoidance for Baxter’s arms.

Turtlesim and Introduction To ROS

TURTLESIM 01/23/2015

LOG ON AND OPEN A TERMINAL WINDOW. This example will use three windows to run the simulation. A forth window will be used to show the use of ROS to get information about the simulation such as the position of the turtle.

Terminal 1 Start ROS

```
tlharmanphd@D125-43873:~$ roscore
```

```
... logging to /home/tlharmanphd/.ros/log/1de04490-a353-11e4-86c8-3417ebbca982/roslaunch-D125-43873-13463.log
```

```
Checking log directory for disk usage. This may take a while.
```

```
Press Ctrl-C to interrupt
```

```
Done checking log file disk usage. Usage is <1GB.
```

```
started roslaunch server http://D125-43873:46355/
```

```
ros_comm version 1.9.55
```

```
SUMMARY
```

```
=====
```

```
PARAMETERS
```

```
* /roscdistro
```

```
* /rosversion
```

```
NODES
```

```
auto-starting new master
```

```
process[rosmaster]: started with pid [13477]
```

```
ROS_MASTER_URI=http://D125-43873:11311/
```

```
setting /run_id to 1de04490-a353-11e4-86c8-3417ebbca982
```

```
process[rosout-1]: started with pid [13490]
```

```
started core service [/rosout]
```

You can ignore the messages for this simulation.

Turtlesim Terminal 2

```
tlharmanphd@D125-43873:~$ rosrun turtlesim turtlesim_node
[ INFO] [1422053853.021652635]: Starting turtlesim with node name /turtlesim
[ INFO] [1422053853.024476555]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.000000]
```

Turtlesim is the package and Turtlesim_node is the executable node that creates the picture with the turtle.

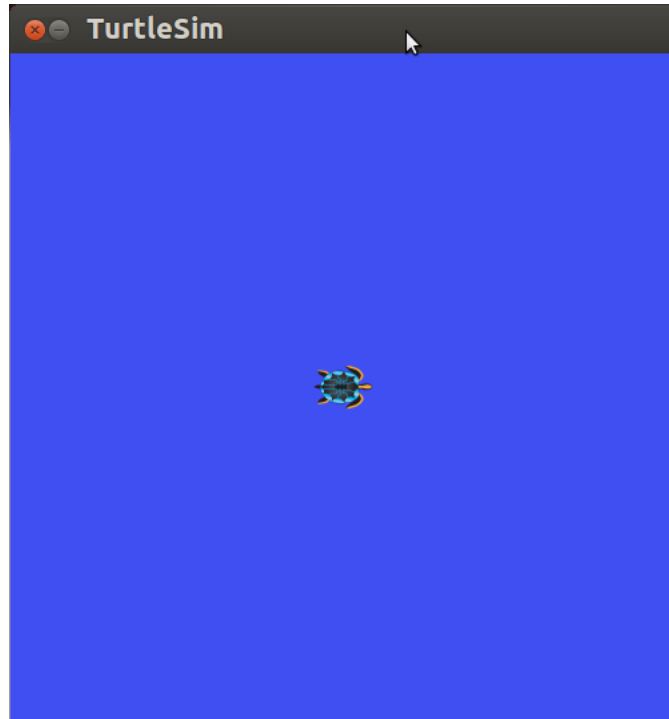


Figure 6 Turtlesim 1 with the Turtle after the node is executed

TURTLESIM WINDOW – TO MOVE TURTLE OPEN NEW WINDOW #3 TO ENABLE KEYBOARD.

TERMINAL 3 Enable the Keyboard

In the third window, we execute a node that allows keyboard control of the turtle.

```
tlharmanphd@D125-43873:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.

Up arrow      Turtle up
Down arrow    Turtle down
Right arrow    Rotate CW
Left arrow     Rotate CCW
```

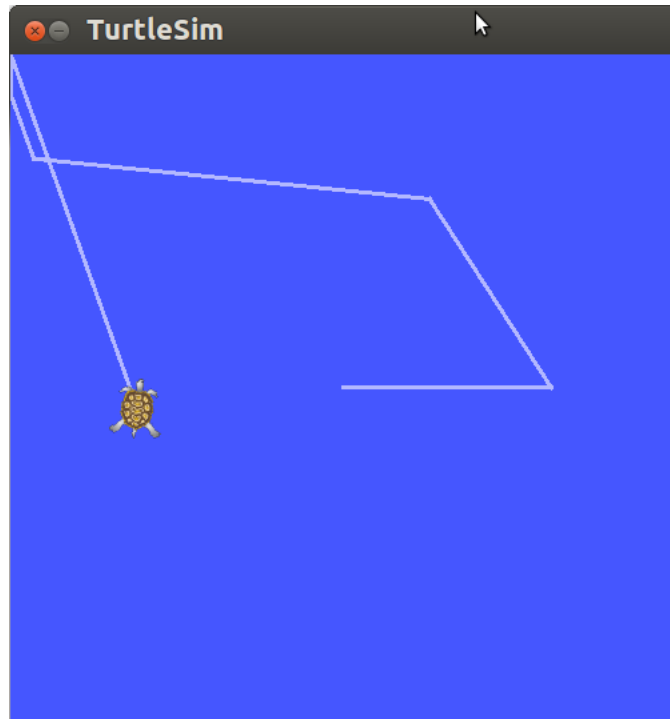


Figure 7 Turtlesim After Moving

TURTLESIM WINDOW AFTER USING TERMINAL 3 WITH KEYBOARD

TERMINAL 4 ROS Nodes, Topics, and Services Using Turtlesim

Before going through this section and especially the tutorials on ros.org, you should have read and understood the material about ROS covered in the *Introduction to Baxter* report by T.L. Harman and Carol Fairchild.

A READ OF THE FIRST FEW CHAPTERS IN BOOKS SUCH AS THE FOLLOWING WILL BE HELPFUL:

The textbook *Learning ROS for Robotics Programming* by Aaron Martinez is useful. The examples are in C++.


A *Gentle Introduction to ROS* by Jason M. O’Kane is very readable and can be downloaded from the site: <http://www.cse.sc.edu/~jokane/agitr/agitr-letter.pdf>
The author’s website is <http://www.cse.sc.edu/~jokane/agitr/>

These other ROS books might be helpful as referenced by O’Kane:

- [ROS by Example](#) by R. Patrick Goebel
- [Learning ROS for Robotics Programming](#)

by Aaron Martinez and Enrique Fernandez. The examples are in C++.

Always be sure to check of any changes in the Ubuntu or ROS distribution. This User’s Guide is written using Ubuntu 12.04 and ROS Groovy as indicated in the *Introduction to Baxter* report by T.L. Harman and Carol Fairchild.

 If you are new to ROS - don’t be impatient. There is a great deal to learn but the Turtlesim example shown here should make things easier.

The ROS official tutorials are at these WEB sites:

<http://wiki.ros.org/turtlesim/Tutorials>

ROS Tutorials Helpful for the Examples to Follow:

- [ROS/Tutorials/UnderstandingNodes](#)
- [ROS/Tutorials/UnderstandingTopics](#)
- [ROS/Tutorials/UnderstandingServicesParams](#)

We start a fourth terminal window to view the information that is available through ROS for the Turtlesim. The commands in that window elicit data while the other windows keep the turtle active. To move the turtle, use window three.

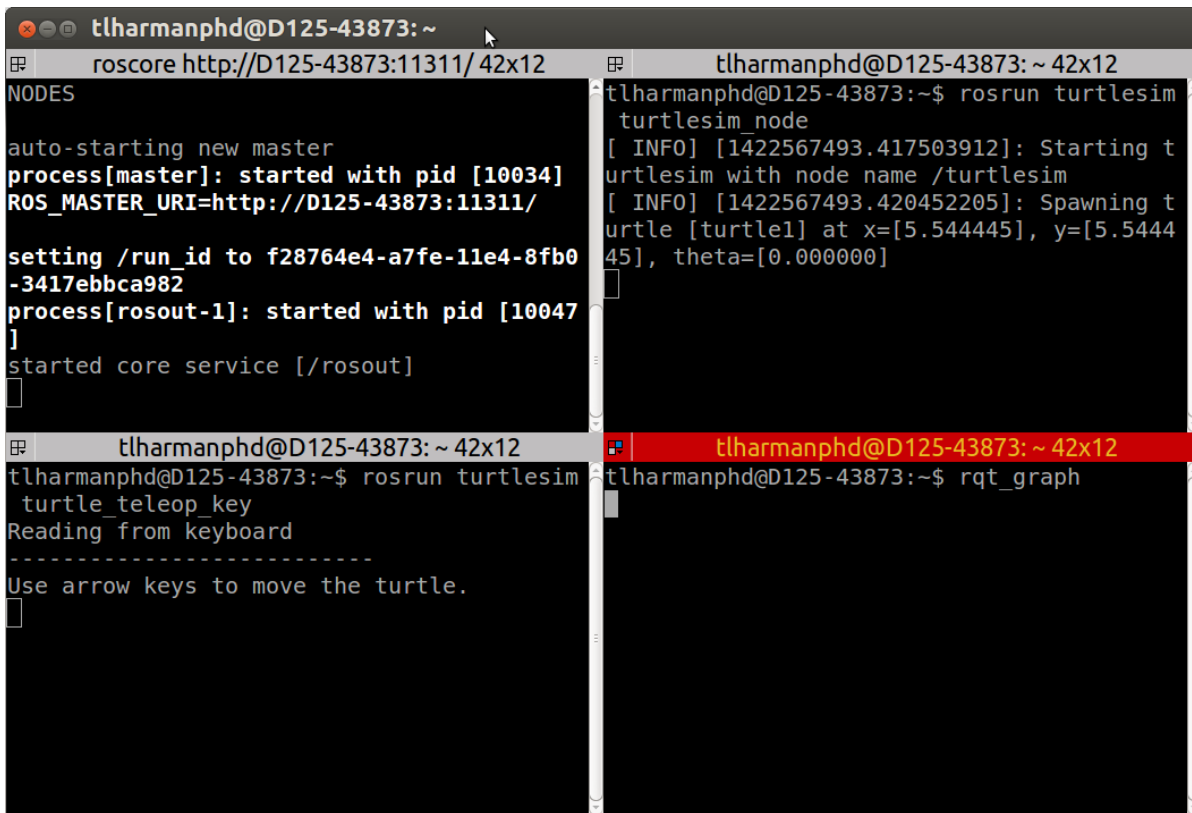


Figure 8 Four Turtlesim Windows using Terminator

The screen with four windows was created using Terminator. It is downloaded in Ubuntu from the Software Center Icon on the launcher: http://en.wikipedia.org/wiki/Ubuntu_Software_Center

The terminator is described at this site: <https://apps.ubuntu.com/cat/applications/terminator/>

1. List the ROS parameters to get information about the ROS nodes. The nodes are generally the executable scripts in ROS.

tlharmanphd@D125-43873:~\$ **roscout**

roscout is a command-line tool for printing information about ROS Nodes.

Commands:

```
roscout ping  test connectivity to node
roscout list  list active nodes
roscout info  print information about node
roscout machine      list nodes running on a particular machine or list machines
roscout kill  kill a running node
roscout cleanup     purge registration information of unreachable nodes
```

Type roscout <command> -h for more detailed usage, e.g. 'roscout ping -h'

```

-----
tlharmanphd@D125-43873:~$ rostopic list -h
Usage: rostopic list
Options:
  -h, --help  show this help message and exit
  -u          list XML-RPC URIs
  -a, --all   list all information
tlharmanphd@D125-43873:~$ rostopic list      (Active Nodes)
/rosout
/teleop_turtle
/turtlesim
tlharmanphd@D125-43873:~$

```

2. Determine what information you can get for the node turtlesim.

```

tlharmanphd@D125-43873:~$ rostopic info /turtlesim      (P44)
-----

```

```

Node [/turtlesim]
Publications:
* /turtle1/color_sensor [turtlesim/Color]
* /rosout [roscpp_msgs/Log]
* /turtle1/pose [turtlesim/Pose]

Subscriptions:
* /turtle1/command_velocity [turtlesim/Velocity]

Services:
* /turtle1/teleport_absolute
* /turtlesim/get_loggers
* /turtlesim/set_logger_level
* /reset
* /spawn
* /clear
* /turtle1/set_pen
* /turtle1/teleport_relative
* /kill

```

contacting node http://D125-43873:52428/ ... (Our lab workstation)

Pid: 13590

Connections:

```

* topic: /rosout
  * to: /rosout
  * direction: outbound
  * transport: TCPROS
* topic: /turtle1/command_velocity
  * to: /teleop_turtle (http://D125-43873:43507/)
  * direction: inbound
    ⑩ transport: TCPROS
    ⑩

```

```
tlharmanphd@D125-43873:~$ rostopic list (Publications and Subscriptions)
/rosout
/rosout_agg
/turtle1/color_sensor
/turtle1/command_velocity
/turtle1/pose
```

`rosout` is the name of the console log reporting mechanism in ROS. It can be thought as comprising several components:

For a little explanation from the ROS wiki:

<http://wiki.ros.org/rosout>

- The ``rosout`` node for subscribing, logging, and republishing the messages.
- The `/rosout` topic
- The `/rosout_agg` topic for subscribing to an aggregated feed
- `rosgraph_msgs/Log` message type, which defines standard fields as well as [verbosity levels](#).

The `rosout` package only provides the `rosout` node.

One important topic is `/turtle1/command_velocity` which will be **published** using the keyboard or by publishing the topic with the `rostopic pub` command as shown later.

DETERMINE DATA

This command shows the data sent by the node to control the turtle. As you move the turtle, the data are updated.

```
tlharmanphd@D125-43873:~$ rostopic echo /turtle1/command_velocity
```

```
---
linear: 2.0          (Use Window 3 to move Turtle)
angular: 0.0
---
linear: 2.0
angular: 0.0
---
linear: -2.0
angular: 0.0
---
linear: 2.0
angular: 0.0
---
```

Services allow nodes to communicate by sending a request and receiving a response. The services can be used in the form `rosservice call <option>` where option for example is `/clear`.


```
tlharmanphd@D125-43873:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/spawn
/teleop_turtle/get_loggers
/teleop_turtle/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
```

We can make the turtle turn in a circle by **publishing** the topic `/turtle1/command_velocity`.

```
tlharmanphd@D125-43873:~$ rostopic pub /turtle1/command_velocity turtlesim/Velocity -r 1 -- 2.0 -1.8
```

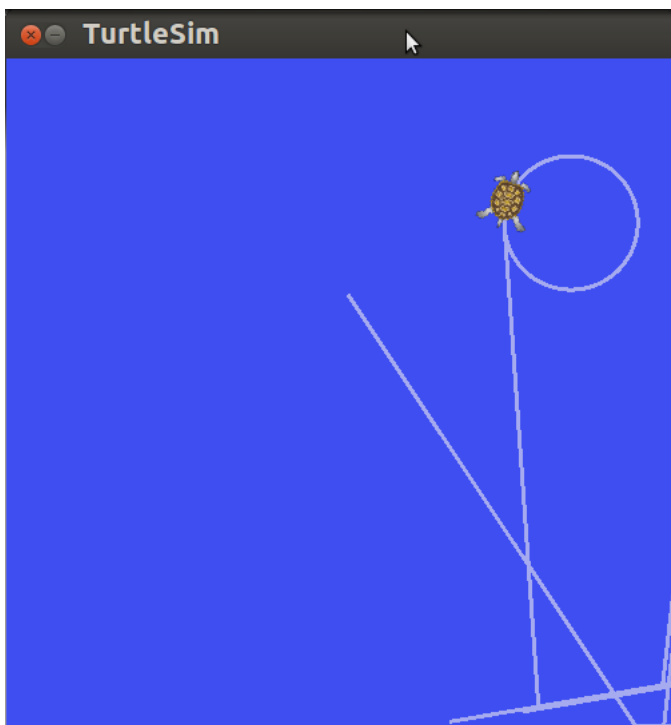


Figure 9 Turtle responds to published topic

The command will publish at a rate (`-r`) of once a second (1 Hz). The topic `/turtle1/command_velocity` is followed by the message type `turtlesim/Velocity` that commands the turtle to turn with linear velocity 2.0 and angular velocity 1.8 according to the ROS tutorial:

<http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics>

As noted before, a turtlesim/Velocity msg has two floating point elements : `linear` and `angular`. In this case, `2.0` becomes the linear value, and `1.8` is the angular value. These arguments are actually in YAML syntax, which is described more in the [YAML command line documentation](#).

When you want to CLEAR THE SCREEN

```
tlharmanphd@D125-43873:~$ rosservice call /clear
```

There is another feature of ROS that is useful for those who wish to see a graphical view of the communication between nodes. We know that `/teleop_turtle` node **publishes** a message on the topic called `/turtle1/command_velocity` and the node `/Turtlesim` **subscribes** to those messages.

This can be shown in a graphical form with the command:

```
tlharmanphd@D125-43873:~$ rqt_graph
```

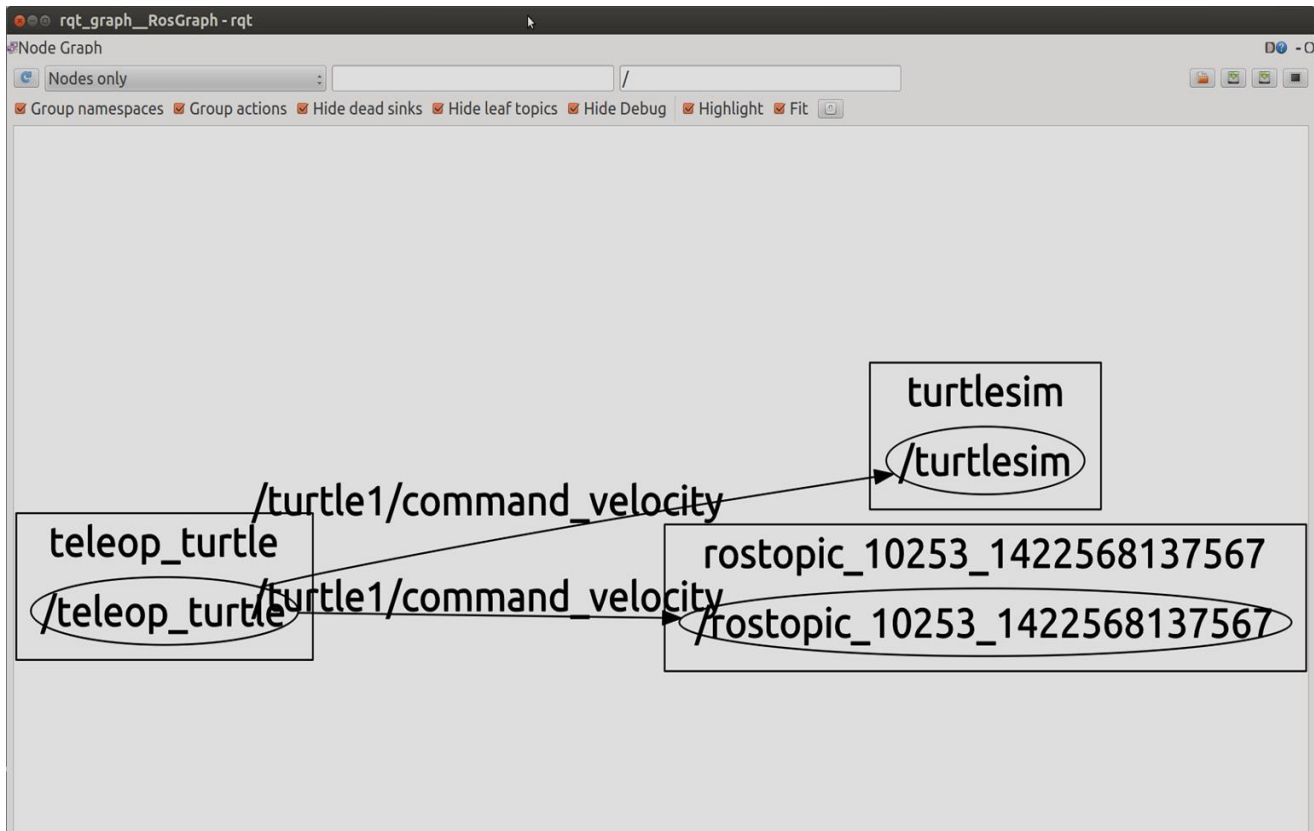


Figure 10 Turtlesim graph showing communication

The advantage of Turtlesim is as follows:

1. Easy to Learn and Use
2. Shows basic ROS capability
3. Can be downloaded with ROS for use on a laptop

Working with Baxter and ROS is considerably more complicated. The software modules are pictured in Appendix I and the ROS files are listed in Appendix III.

MoveIt

From the official MoveIt site:

“MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains.” View the video on that site.

<http://moveit.ros.org/>

Here is an introduction to ROS and MoveIt together:

<https://www.youtube.com/watch?v=eMIGV94c5WU>

The Rethink tutorial shows how to download and use MoveIt with Baxter:

<https://github.com/RethinkRobotics/sdk-docs/wiki/MoveIt-Tutorial>

Rethink Robotics has a good video showing the use of MoveIt:

<https://www.youtube.com/watch?v=1Zdkwym42P4>

To use MoveIt:

In First terminal window:

- 1.Go to ros_ws, Execute baxter.sh
- 2.Enable Baxter and run the joint_trajectory_action_server

```
tlharmanphd@D125-43873:~$ cd ~/ros_ws
tlharmanphd@D125-43873:~/ros_ws$ ./baxter.sh
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ rosrun baxter_tools
enable_robot.py -e
  [INFO] [WallTime: 1422563798.218496] Robot Enabled
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ rosrun baxter_interface
joint_trajectory_action_server.py
  Initializing node...
  Initializing joint trajectory action server...
  Running. Ctrl-c to quit
```

In Second terminal window: Go to ros_ws, run baxter.sh again and launch MoveIt

```
tlharmanphd@D125-43873:~/ros_ws$ ./baxter.sh
```

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ roslaunch  
baxter_moveit_config_demo_baxter.launch
```

```
... logging to /home/tlharmanphd/.ros/log/adfe730a-a7e3-11e4-bae5-000af72ca0bb/ros-launch-  
D125-43873-7674.log
```

```
Checking log directory for disk usage. This may take awhile.
```

```
Press Ctrl-C to interrupt
```

```
Done checking log file disk usage. Usage is <1GB.
```

```
started roslaunch server http://172.29.64.201:60551/
```

After a great deal of information look for the line

All is well! Everyone is happy! You can start planning now!

The roslaunch command causes a number of nodes to execute.

The Rviz gui will then open showing Baxter with interactive markers:

MoveIT first screen with Baxter

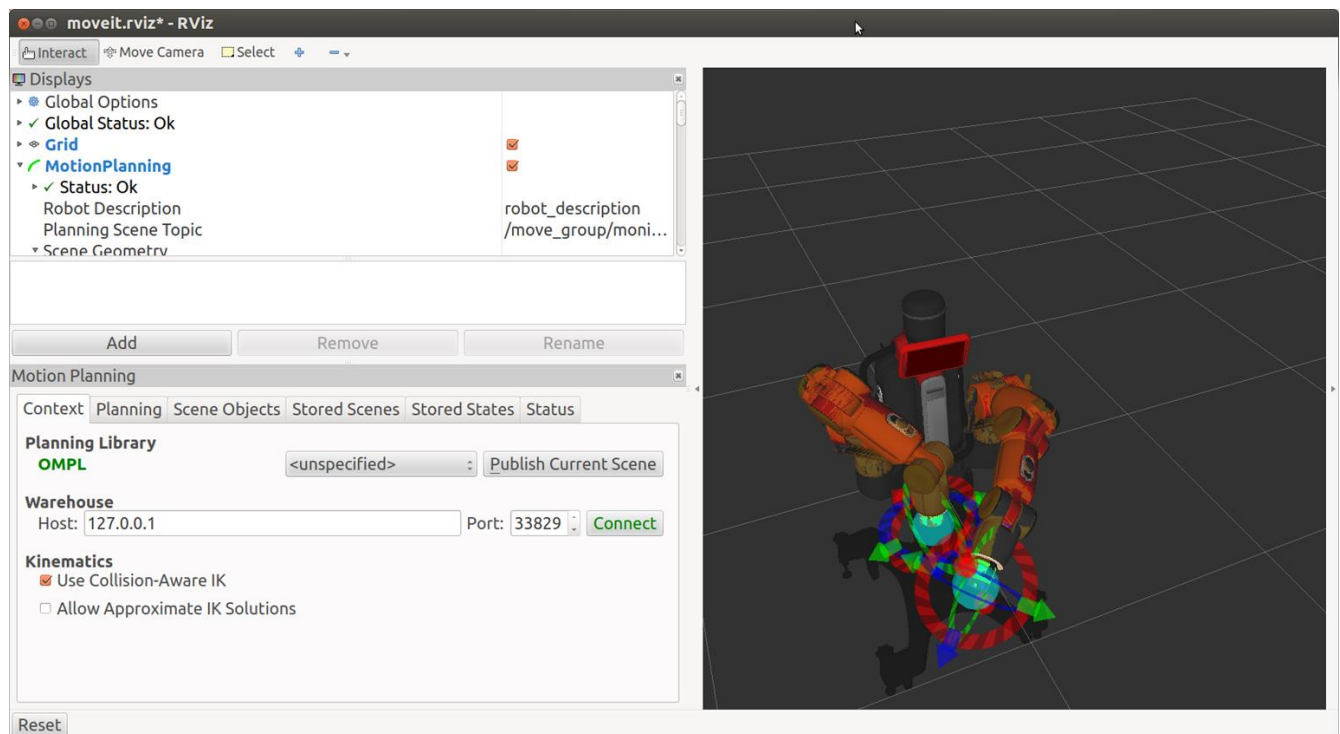


Figure 11 MoveIt and Baxter Initial View

SELECT THE PLANNING TAB and move the simulated arms using the arrows and rings.

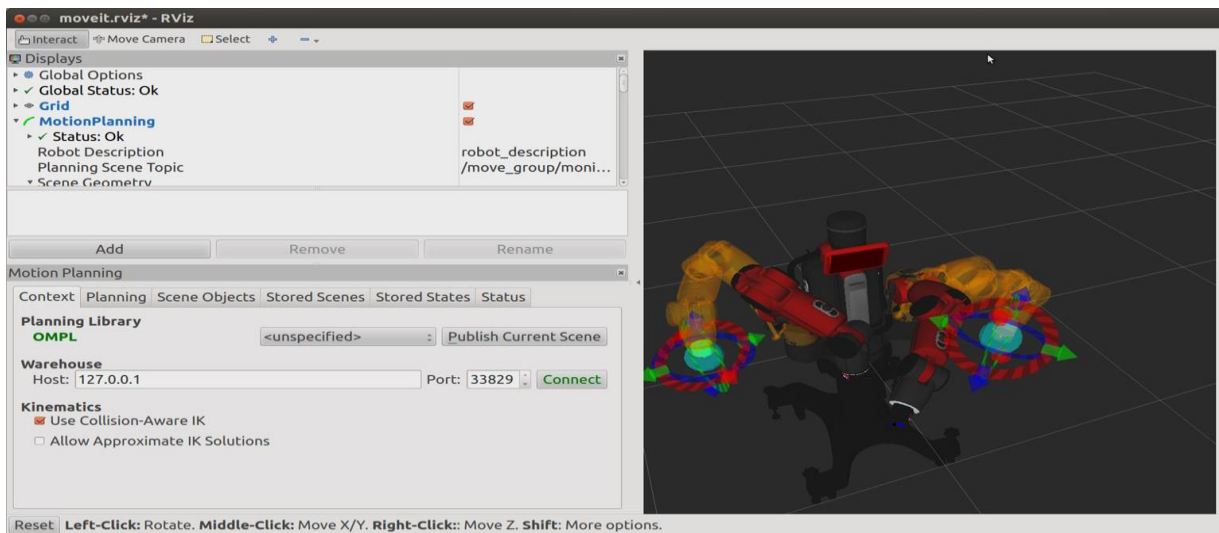


Figure 12 Simulator with Initial and Desired Position of Arms

Viewing Baxter with Red arms as current position, Orange as planned position

Now using Execute, Baxter's physical arms will move to the simulated position.

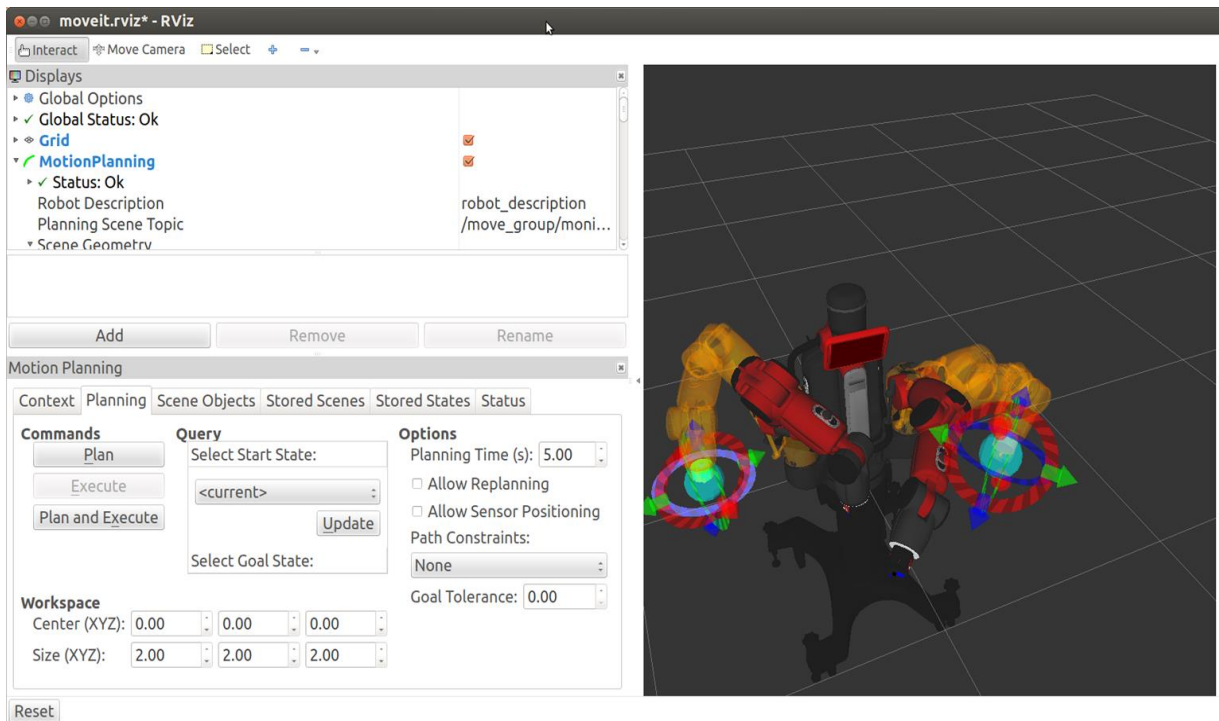


Figure 13 Baxter Simulator Ready for Execution of Moves

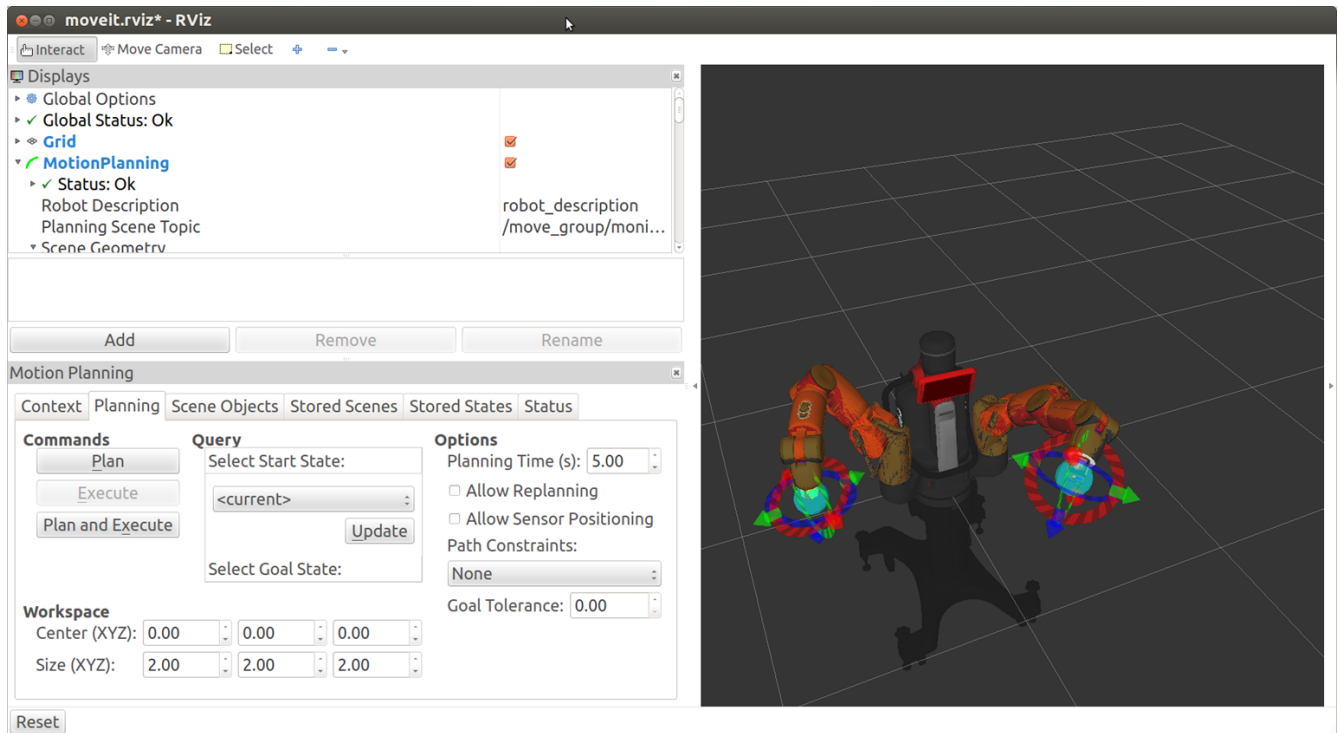


Figure 14 Baxter Simulation Showing Final Position of Arms

EXECUTE - Moves Baxter's arms to positions as in the simulation.

Gazebo

Gazebo is used to simulate many robots and Baxter is one of them.

Baxter Simulation with Gazebo. To download the Baxter Simulator used with Gazebo, it requires a GitHub user name and permission from Rethink Robotics.

TURN ON BAXTER AND LOG IN.

The option `sim` was added to the `baxter` shell.

In the first terminal window:

```
fairchildc@D125-43873:~$ ros_ws
```

```
fairchildc@D125-43873:/home/tlharmanphd/ros_ws$ ./baxter.sh sim
```

```
[baxter - http://localhost:11311] fairchildc@D125-43873:/home/tlharmanphd/ros_ws$ roslaunch  
baxter_gazebo baxter_world.launch
```

```
... logging to /home/fairchildc/.ros/log/f75eaa0-a647-11e4-b327-3417ebbca982/roslaunch-  
D125-43873-8765.log
```

```
Checking log directory for disk usage. This may take awhile.
```

```
Press Ctrl-C to interrupt
```

```
Done checking log file disk usage. Usage is <1GB.
```

```
started roslaunch server http://172.29.64.201:44979/
```

```
SUMMARY
```

```
=====
```

```
PARAMETERS
```

```
* /grav_left_name
```

```
* /grav_right_name
```

```
* /left_tip_name
```

```
...
```

Baxter Simulation is ready when you see the following:

```
[ INFO] [1422380584.229118484, 529.025000000]: Robot is enabled
```

```
[ INFO] [1422380584.229174028, 529.025000000]: right_joint_position_controller was started  
and right_joint_velocity_controller and right_joint_effort_controller were stopped  
succesfully
```

```
[ INFO] [1422380584.229194997, 529.025000000]: Gravity compensation was turned on
```

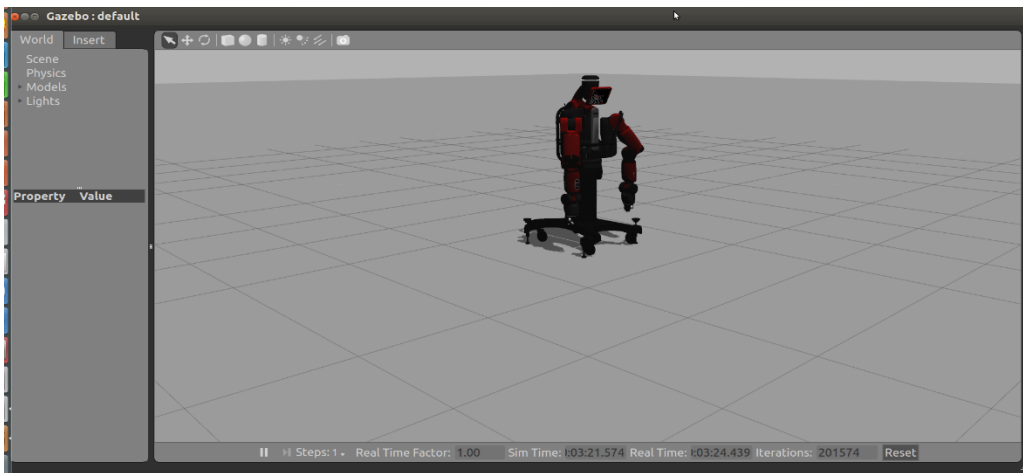



Figure 15 Gazebo Simulation of Baxter before Enabling Baxter

In a second terminal window:

```

fairchildc@D125-43873:~$ ros_ws
fairchildc@D125-43873:/home/tlharmanphd/ros_ws$ ./baxter.sh sim
[baxter - http://localhost:11311] fairchildc@D125-43873:/home/tlharmanphd/ros_ws$ rosrund
baxter_tools enable_robot.py -e
      [INFO] [WallTime: 1422380502.352324] [447.269000] Robot Enabled
[baxter - http://localhost:11311] fairchildc@D125-43873:/home/tlharmanphd/ros_ws$ . run_baxter
untuck

```

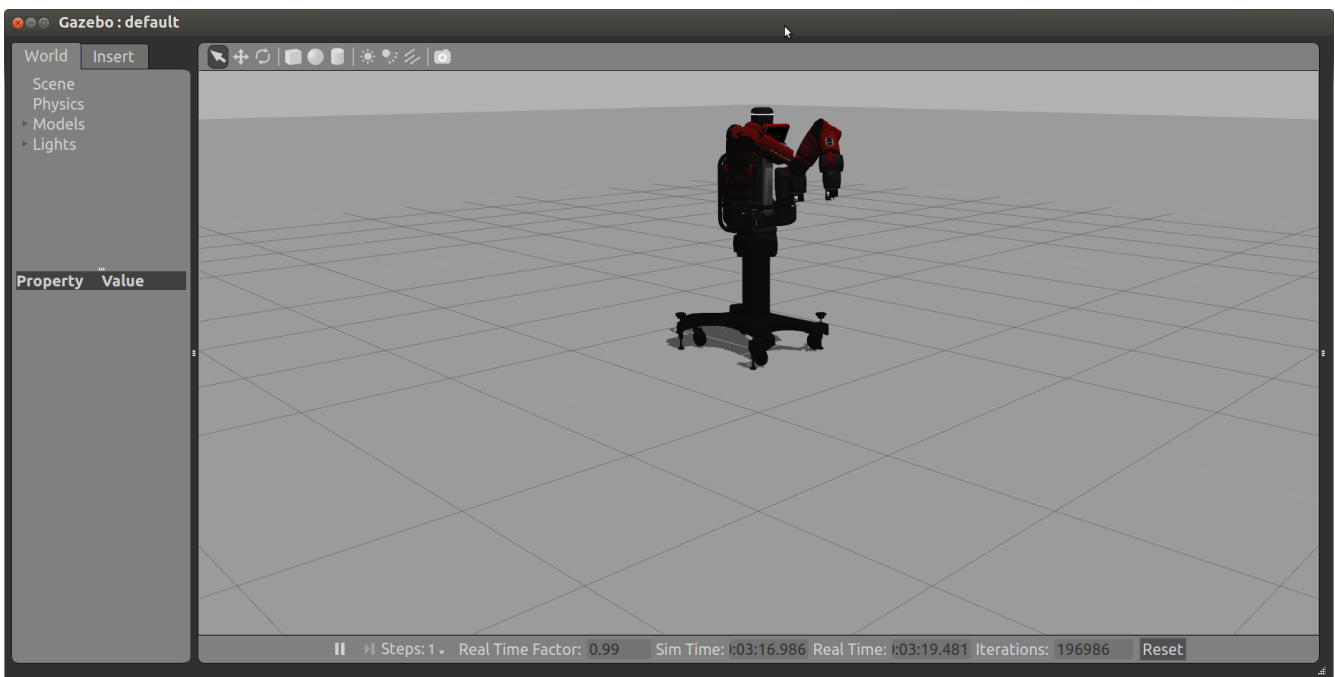


Figure 16 Baxter Enabled in Simulation

```
Today is Tue Jan 27 11:43:03 CST 2015
[INFO] [WallTime: 1422380583.925996] [0.000000] Untucking arms
[INFO] [WallTime: 1422380584.016426] [528.812000] Moving head to neutral position
[INFO] [WallTime: 1422380584.016611] [528.812000] Untucking: Arms already Untucked; Moving
to neutral position.
[INFO] [WallTime: 1422380585.581263] [530.374000] Finished tuck
[baxter - http://localhost:11311] fairchildc@D125-43873:/home/tlharmanphd/ros_ws$
```

The command `$ roslaunch baxter_tools tuck_arms.py -u` could have been used but here we used Carol's script `. run_baxter` discussed previously and in Appendix IV.

TRY AN EXAMPLE TO SEE BAXTER MOVE IN SIMULATION:

```
$ roslaunch baxter_examples joint_velocity_wobbler.py
```

APPENDIX I BAXTER RETHINK EXAMPLES

Movement

Joint Position Waypoints Example - The basic example for joint position moves. Hand-over-hand teach and recording a number of joint position waypoints. These waypoints will then be played back indefinitely upon completion.

Joint Position Keyboard Example - This example demonstrates numerous joint position control.

Joint Position Example - Joystick, keyboard and file record/playback examples using joint position control of Baxter's arms.

Joint Torque Springs Example - Joint torque control example applying virtual spring torques.

Joint Velocity Wobbler Example - Simple demo that moves the arm with sinusoidal joint velocities.

Joint Velocity Puppet Example - Demo which mirrors moves of one arm on the other in Zero-G.

Inverse Kinematics Service Example - Basic use of Inverse Kinematics solver service.

Simple Joint Trajectory Example - Simple demo using the joint trajectory interface.

Joint Trajectory Playback Example - Trajectory playback using the joint trajectory interface.

Head Movement Example - Simple demo moving and nodding the head.

Gripper Example - Joystick and Keyboard control for the grippers.

Input and Output

Camera Control Example - Demonstrates usage for listing, opening, and closing the available cameras.

View Cameras Example - Simple tool for viewing camera feed on development machine.

I/O Example - Flash the lights on the digital outputs.

These examples are shown as part of Baxter's software in Figure 17.

The Figure shows the elements of software loaded on the workstation to download software or operate Baxter. Notice the Baxter Interface, Examples and Tools software modules. The figure is for the SDK version 0.7 and may not be up to date with the latest SDK software from Rethink Robotics.

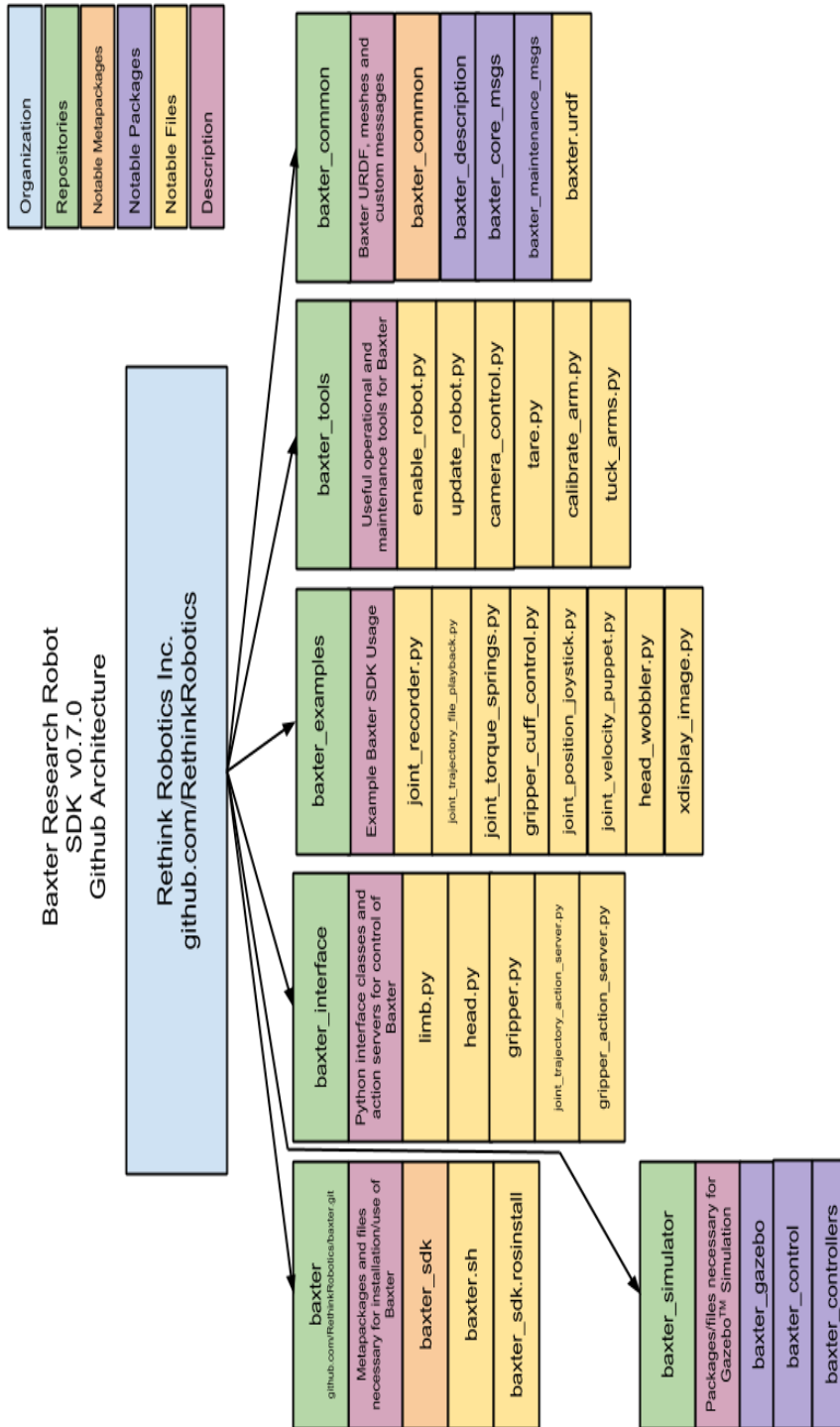


Figure 17 Baxter Software

Examples with Links From FOUNDATIONS 8/30/2014

<http://sdk.rethinkrobotics.com/wiki/Examples>

Here are links to the example pages:

- [Wobbler Example](#) Use wobbler as an example of controlling Baxter using joint velocity control. Arms “wobble”
- [Joint Trajectory Playback Example](#) Enable the robot joint trajectory interface, parse a file created using the joint position recorder example, and send a the resulting joint trajectory to the action server.
- [Puppet Example](#) Use puppet as an example of controlling Baxter using joint velocity control. One arm follows the other in VELOCITY.
- [Simple Joint Trajectory Example](#) Enable the robot joint trajectory interface and send a simple joint trajectory for Baxter to follow.
- [Joint Torque Springs Example](#) This example shows joint torque control usage. After moving to neutral, the robot will enter torque control mode, applying torques representing virtual springs holding the joints to their start position.
- [Gripper Example](#) Use gripper control as an example of controlling Baxter's grippers.
- [Head Movement Example](#) The Head "Wobbler" Example randomly moves the head to demonstrate using the head pan and nod interfaces.
- [IK Service Example](#) The IK Test example shows the very basics of calling the on-robot Inverse-Kinematics (IK) Service to obtain a joint angles solution for a given endpoint Cartesian point & orientation.
- [Input Output Example](#) The 'input_output' examples provide simple demonstrations of using the DigitalIO and AnalogIO components, in correspondence with the python interfaces included in the 'baxter_interface' code.

APPENDIX II Summary of Ubuntu and ROS commands

UNITY		TERMINAL	
Dash Help ? F1 –	Desktop Guide Works with many applications	cmd --help help cmd man cmd whatis cmd info cmd	cp –help help pwd man echo whatis grep info grep
FILES DIRECTORIES			
Home Folder -File System	Show Root/home/home-user	ls -la	Show all and long-permissions, etc.
RightClick Copy Then Paste		cp <options> file1 file2	Copy -r does files in directories
EDIT			
LibreOffice	Save as txt if a program- change suffix	gedit file	Move up to menu bar and File:SaveAs
SYSTEM			
Sysinfo in Dash SystemSettings-Launcher or on Right of menu bar	(Installed) -Hardware Appearance, HW, Network...,details, User Accounts	lspci -v	More than you need
NETWORK			
SystemSettings Network	172.29.64.201	ifconfig	

APPENDIX III ROS Workspace Directories

/home/tlharmanphd/ros_ws/baxter.sh
/home/tlharmanphd/ros_ws/baxter_common
/home/tlharmanphd/ros_ws/build
/home/tlharmanphd/ros_ws/devel

/home/tlharmanphd/ros_ws/src
/home/tlharmanphd/ros_ws/src/activerobots
/home/tlharmanphd/ros_ws/src/baxter
/home/tlharmanphd/ros_ws/src/baxter_common
/home/tlharmanphd/ros_ws/src/baxter_examples
/home/tlharmanphd/ros_ws/src/baxter_interface
/home/tlharmanphd/ros_ws/src/baxter_simulator
/home/tlharmanphd/ros_ws/src/baxter_tools
/home/tlharmanphd/ros_ws/src/control_toolbox
/home/tlharmanphd/ros_ws/src/gazebo_ros_pkgs

/home/tlharmanphd/ros_ws/src/moveit_robots
/home/tlharmanphd/ros_ws/src/moveit_robots/clam_moveit_config
/home/tlharmanphd/ros_ws/src/moveit_robots/iri_wam_moveit_config
/home/tlharmanphd/ros_ws/src/moveit_robots/r2_moveit_generated

/home/tlharmanphd/ros_ws/src/realtime_tools

/home/tlharmanphd/ros_ws/src/ros_control
/home/tlharmanphd/ros_ws/src/ros_control/controller_interface
/home/tlharmanphd/ros_ws/src/ros_control/controller_manager
/home/tlharmanphd/ros_ws/src/ros_control/controller_manager_msgs
/home/tlharmanphd/ros_ws/src/ros_control/controller_manager_tests
/home/tlharmanphd/ros_ws/src/ros_control/docs
/home/tlharmanphd/ros_ws/src/ros_control/hardware_interface
/home/tlharmanphd/ros_ws/src/ros_control/joint_limits_interface
/home/tlharmanphd/ros_ws/src/ros_control/ros_control
/home/tlharmanphd/ros_ws/src/ros_control/transmission_interface

/home/tlharmanphd/ros_ws/src/ros_controllers
/home/tlharmanphd/ros_ws/src/ros_controllers/effort_controllers
/home/tlharmanphd/ros_ws/src/ros_controllers/force_torque_sensor_controller
/home/tlharmanphd/ros_ws/src/ros_controllers/forward_command_controller
/home/tlharmanphd/ros_ws/src/ros_controllers/imu_sensor_controller
/home/tlharmanphd/ros_ws/src/ros_controllers/joint_state_controller
/home/tlharmanphd/ros_ws/src/ros_controllers/joint_trajectory_controller
/home/tlharmanphd/ros_ws/src/ros_controllers/position_controllers
/home/tlharmanphd/ros_ws/src/ros_controllers/ros_controllers
/home/tlharmanphd/ros_ws/src/ros_controllers/velocity_controllers

/home/tlharmanphd/ros_ws/src/xacro

APPENDIX IV Carol's Script for .run_baxter

01/23/15 .run_baxter

```
[baxter - http://172.29.64.200:11311] tlharmanphd@D125-43873:~/ros_ws$ .run_baxter h
Today is Fri Jan 23 15:27:32 CST 2015
run_baxter commands:
```

enable, disable, state, reset, stop

tuck, untuck

arms_keyboard, record <filename>, playback <filename>

springs <right or left>, arms_wobbler, puppet <right or left>

ik <right or left>, joint_trajectory <right or left>

camera open <right, left or head> res <wide, medium or narrow>

camera close <right, left or head>

head_wobbler, gripper_keyboard, head_display <filename>

digital_io, analog_io

The script runs under Ubuntu/BASH and calls various Python language scripts from the Baxter ROS packages.


```

# This script combines all of the Baxter examples into equivalent
# run_baxter commands.

#!/bin/bash
echo "Today is `date`"

case "$1" in

    # enable_robot from baxter_tools
    enable | Enable | ENABLE )
        rosrun baxter_tools enable_robot.py -e
        ;;
    disable | Disable | DISABLE )
        rosrun baxter_tools enable_robot.py -d
        ;;
    state | State | STATE )
        rosrun baxter_tools enable_robot.py -s
        ;;
    reset | Reset | RESET )
        rosrun baxter_tools enable_robot.py -r
        ;;
    stop | Stop | STOP )
        rosrun baxter_tools enable_robot.py -S
        ;;

    # The following options will open a new terminal window and execute the command.

    # tuck_arms from baxter_tools
    tuck | Tuck | TUCK )
        rosrun baxter_tools tuck_arms.py -t
#       xterm -hold -e "rosrun baxter_tools tuck_arms.py -t"
        ;;
    untuck | Untuck | UNTUCK )
        rosrun baxter_tools tuck_arms.py -u
#       xterm -hold -e "rosrun baxter_tools tuck_arms.py -u"
#       #gnome-terminal -e "rosrun baxter_tools tuck_arms.py -u" -- did not work
        ;;

    # joint commands from baxter_examples
    arms_keyboard | ARMS_KEYBOARD )
        xterm -hold -e "rosrun baxter_examples joint_position_keyboard.py"
        ;;

    record | RECORD )
        if [ -n "${2}" ]; then      # check if second argument is passed
            xterm -hold -e "rosrun baxter_examples joint_recorder.py -f $2"
        else

```

```

    echo "No Filename given."
fi
;;

playback | PLAYBACK )
if [ -r "${2}" ]; then      # check that file exists and is readable
    xterm -hold -e "roslaunch baxter_examples joint_position_file_playback.py -f $2"
else
    echo "No Filename given or file does not exist."
fi
;;

# joint torque command from baxter_examples
springs | Springs | SPRINGS )
if [ "$2" == left -o "$2" == right ]; then
    xterm -hold -e "roslaunch baxter_examples joint_torque_springs.py -l $2"
else
    echo "Specify right or left."
fi
;;

arms_wobbler | Arms_Wobbler | ARMS_WOBBLER | \
arm_wobbler | Arm_Wobbler | ARM_WOBBLER | wobbler )
xterm -hold -e "roslaunch baxter_examples joint_velocity_wobbler.py"
;;

puppet | Puppet | PUPPET )
if [ "$2" == left -o "$2" == right ]; then
    xterm -hold -e "roslaunch baxter_examples joint_velocity_puppet.py -l $2"
else
    echo "Specify right or left."
fi
;;

# call on-board Inverse Kinematics (IK) service to obtain joint angle solution
# for a given endpoint Cartesian point & orientation
ik | IK )
if [ "$2" == left -o "$2" == right ]; then
    xterm -hold -e "roslaunch baxter_examples ik_service_client.py -l $2"
else
    echo "Specify right or left."
fi
;;

# enable robot joint trajectory interface using joint trajectory controller
# another terminal session can execute a client to send a joint trajectory
# to the right or left arm
joint_traj* | Joint_Traj* | JOINT_TRAJ* )

```

```

if [ "$2" == left -o "$2" == right ]; then
    xterm -hold -e "roslaunch baxter_interface joint_trajectory_action_server.py" &
    roslaunch baxter_examples joint_trajectory_client.py -l $2
else
    echo "Specify right or left."
fi
;;

# head_wobbler randomly nods and tilts head
head_wobbler | Head_Wobbler | HEAD_WOBBLER )
    xterm -hold -e "roslaunch baxter_examples head_wobbler.py"
;;

# gripper_keyboard uses keyboard to control grippers
gripper_keyboard | Gripper_Keyboard | GRIPPER_KEYBOARD )
    xterm -hold -e "roslaunch baxter_examples gripper_keyboard.py"
;;

#-----
# camera control tool
camera | Camera | CAMERA )

    roslaunch baxter_tools camera_control.py -c right_hand_camera
    roslaunch baxter_tools camera_control.py -c left_hand_camera
    roslaunch baxter_tools camera_control.py -c head_camera

    shift
    key="$1"

    case $key in
        open | Open | OPEN )

            shift
            roslaunch baxter_tools camera_control.py -c right_hand_camera
            roslaunch baxter_tools camera_control.py -c left_hand_camera
            roslaunch baxter_tools camera_control.py -c head_camera
            RESOLUTION=1280x800          # default resolution

            case "$1" in
                right | Right | RIGHT )
                    CAMERA=right_hand_camera
                    ;;
                left | Left | LEFT )
                    CAMERA=left_hand_camera
                    ;;
                head | Head | HEAD )
                    CAMERA=head_camera
                    ;;
            esac
        ;;
    esac

```

```

*)
    echo "Camera not specified."
    echo "open <right, left or head>"
    exit
;;
esac
shift

# if there are more parameters on the command line do the next steps
while [[ $# > 1 ]];
do

case "$1" in
    res | Res | RES )

        shift

        case "$1" in
            wide | Wide | WIDE )
                RESOLUTION=1280x800
                ;;
            medium | Medium | MEDIUM )
                RESOLUTION=480x300
                ;;
            narrow | Narrow | NARROW )
                RESOLUTION=320x200
                ;;
            *)
                echo "Resolution not specified."
                echo "res <wide, medium or narrow>"
                exit
                ;;
            # for case *
        esac
        ;;
        # for case res
    esac
done
        # end of while loop

roslaunch baxter_tools camera_control.py -o "${CAMERA}" -r "${RESOLUTION}"

# camera image can be displayed on image_view or rviz
# comment out the option you do not want
roslaunch image_view image_view image:=/cameras/"${CAMERA}"/image
#
    xterm -hold -e "roslaunch rviz rviz"

;;
        # for case open

# close camera
close | Close | CLOSE )

```

```

shift

case "$1" in
  right | Right | RIGHT )
    CAMERA=right_hand_camera
    ;;
  left | Left | LEFT )
    CAMERA=left_hand_camera
    ;;
  head | Head | HEAD )
    CAMERA=head_camera
    ;;
  * )
    echo "Camera not specified."
    echo "close <right, left or head>"
    exit
    ;;          # for case *
esac
roslaunch baxter_tools camera_control.py -c "${CAMERA}"
;;          # for case close

* ) echo "camera commands:"
    echo "open <right, left or head> res <wide, medium or narrow>"
    echo "close <right, left or head>"
    ;;          # for case *
esac
;;          # for case camera

#-----

# No new terminal window to execute these commands:

# xdisplay_image
# Display an image (e.g. .png or .jpg) to Baxter's head display. Baxter
# display resolution is 1024 x 600 pixels.
head_display | Head_Display | HEAD_DISPLAY )
  if [ -n "${2}" ]; then
    roslaunch baxter_examples xdisplay_image.py --f $2
  else
    echo "No Filename given."
  fi
  ;;

# digital_io
# This will blink the LED on the Left Navigator on and then off while
# printing the status before and after.
digital_io | Digital_IO | DIGITAL_IO )

```

```

    rosrn baxter_examples digital_io_blink.py
    ;;

# analog_io
# This will run the robot's fans from 0 to 100 and back down again in
# increments of 10.
analog_io | Analog_IO | ANALOG_IO )
    rosrn baxter_examples analog_io_rampup.py
    ;;

* ) echo "run_baxter commands:"
    echo "enable, disable, state, reset, stop"
    echo "tuck, untuck"

    echo "arms_keyboard, record <filename>, playback <filename>"
    echo "springs <right or left>, arms_wobbler, puppet <right or left>"
    echo "ik <right or left>, joint_trajectory <right or left>"

    echo "camera open <right, left or head> res <wide, medium or narrow>"
    echo "camera close <right, left or head>"

    echo "head_wobbler, gripper_keyboard, head_display <filename>"
    echo "digital_io, analog_io"
    ;;
esac

#exit 0 -- using exit will end communication with Baxter --

```

(run_baxter in fairchildc/catkin_ws 01/23/15)