

INTRODUCTION TO BAXTER

**BAXTER**



**UHCL  
ROBOTICS**

## BAXTER INTRODUCTION

Baxter is a two-armed robot purchased for the “Robotics and Control System Laboratory” at the University of Houston Clear Lake (UHCL). Baxter is housed in Delta 125 which will be referred to in this report as “Baxter’s Lab”. The robot was built by Rethink Robotics and Baxter and its applications are described on the web site:

<http://www.rethinkrobotics.com/>

The version of Baxter in Baxter’s lab is the *research version* which can be controlled via a workstation used to create programs and execute them on Baxter. Another version of Baxter is the *manufacturing version* that Rethink sells is trained by “showing” with no programming needed. For example, an operator moves Baxter’s arms and records positions and then replays the motion.

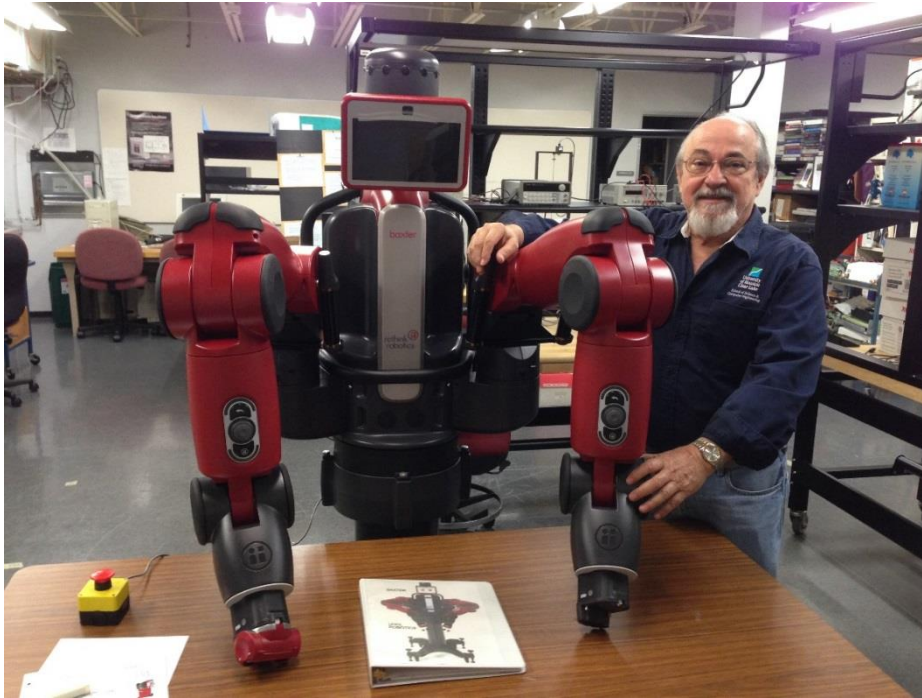


Figure 1 Tom and Baxter in UHCL Lab D125

This report describes the software and hardware elements of the research Baxter system including the workstation and its software and Baxter’s Network. It is NOT a User’s Manual for those wishing to run applications on Baxter. Another report *Baxter User Guide* will describe the use of Baxter and its standard applications called “Rethink Baxter Examples” provided by Rethink Robotics. Another report *Baxter Guide for Advanced Users* will describe how to write Baxter programs or scripts. However, anyone wishing to use Baxter should have a basic understand of the material in this present report. 4/17/2015

# 1 CONTENTS ---- DON'T TRUST THE PAGE NUMBERS

---

<b>BAXTER INTRODUCTION</b> .....	2
<b>Baxter's Team (November 2014)</b> .....	4
<b>INTRODUCTION</b> .....	5
Table 1. Versions of Baxter's Hardware and Development System components .....	5
Table 2. Versions of Software supporting Baxter .....	5
<b>BAXTER'S SYSTEM</b> .....	6
<b>BAXTER'S HARDWARE</b> .....	7
<b>Baxter's Internal Computer</b> .....	10
<b>Baxter's 7DOF Arms</b> .....	11
<b>Zero-G Mode</b> .....	12
<b>References for the Arms and the Series Elastic Actuators:</b> .....	12
<b>Baxter's Electric Gripper</b> .....	13
<b>Baxter's Camera and Sensors</b> .....	14
<b>SDK SOFTWARE FOR BAXTER</b> .....	16
<b>BAXTER SIMULATORS</b> .....	17
<b>MoveIt</b> .....	17
<b>Baxter Simulator Gazebo</b> .....	18
<b>V-REP (Virtual Robot Experimentation Platform)</b> .....	19
<b>ROBOT OPERATING SYSTEM (ROS)</b> .....	20
<b>ROS Terms</b> .....	21
Table 3 ROS Terms .....	21
<b>ROS Tutorials and Books</b> .....	22
<b>ROS Workspace</b> .....	22
<b>UBUNTU AND BASH</b> .....	24
<b>Unity Interface</b> .....	25
<b>Terminal Interface</b> .....	27
<b>What is BASH ?</b> .....	28
<b>CONCLUSIONS</b> .....	32
<b>APPENDIX I Baxter Specifications</b> .....	33
<b>APPENDIX II Unity and Terminal Commands</b> .....	37

## *List of Figures*

<a href="#">Figure 1 Tom and Baxter in UHCL Lab D125</a> .....	1
<a href="#">Figure 2 Angu and Harsha</a> .....	4
<a href="#">Figure 3 Carol and Baxter</a> .....	4
<a href="#">Figure 4 Overall Architecture of Baxter</a> .....	6
<a href="#">Figure 5 Baxter's Physical Structure</a> .....	7
<a href="#">Figure 6 Baxter's External Connections</a> .....	8
<a href="#">Figure 7 Side View of Baxter's Workspace</a> .....	9
<a href="#">Figure 8 Baxter's Internal Circuits</a> .....	10
<a href="#">Figure 9 Baxter's Arm and Joint Designations</a> .....	11
<a href="#">Figure 10 Baxter's Cuff to Enable Zero-G mode</a> .....	12
<a href="#">Figure 11 Baxter's Electric Gripper</a> .....	13
<a href="#">Figure 12 Baxter's camera and IR Sensor in the Cuff</a> .....	14
<a href="#">Figure 13 Baxter's SDK Software</a> .....	16
<a href="#">Figure 14 MoveIt Simulation of Baxter</a> .....	17
<a href="#">Figure 15 Baxter Simulator</a> .....	18
<a href="#">Figure 16 VREP Simulation of Baxter</a> .....	19
<a href="#">Figure 17 Example ROS nodes from Clearpath Robotics</a> .....	22
<a href="#">Figure 18 Getting Started with Ubuntu 12.04</a> .....	24
<a href="#">Figure 19 Unity Desktop</a> .....	25
<a href="#">Figure 20 Rethink Robotics Learning Page</a> .....	30

**Baxter's Team (November 2014)**

Abeysekera, Krishani <Abeysekera@uhcl.edu>;

Carol Fairchild <car.fairchild@gmail.com>;

Findler, Michael <Findler@UHCL.edu>;

Harman, Tom <harman@uhcl.edu>; McKay, Charles [mcKay@uhcl.edu](mailto:mcKay@uhcl.edu)

Miguel Rosales ([RosalesM1720@UHCL.edu](mailto:RosalesM1720@UHCL.edu));



*Figure 2 Angu and Harsha*

*Angusundaresk Krishnakumar (KrishnakumarA2111@UHCL.edu);*

Patil, Harsha ([PatilH3112@UHCL.edu](mailto:PatilH3112@UHCL.edu))



*Figure 3 Carol and Baxter*

## INTRODUCTION

Baxter is a two-armed robot from Rethink Robotics in Boston, MA. The research version that we have at UHCL consists of a number of elements. This report will briefly describe each element and its use to control Baxter. Table 1 summarizes the components of a Baxter system.

Baxter's Hardware	April 2015 version	Robot with two 7DOF arms and various sensors such as sonar and cameras.
WorkStation SDK software	v1.1.0	Workstation with Development System (SDK), Rethink Baxter Examples and APIs. IP: 172.29.64.201
Baxter Simulator		Gazebo and MoveIt simulate Baxter.
Baxter Software	v1.1.0	Baxter's internal software.
Baxter Network IP Address		Baxter - <a href="http://172.29.64.200:11311">http://172.29.64.200:11311</a>

Table 1. Versions of Baxter's Hardware and Development System components

ITEM	VERSION	USE
Operating System (OS)	Ubuntu 14.04.2 LTS Trusty Tahr	System Administrator – Create user accounts and download and maintain software. Users – file management and communication with Baxter through Terminal commands.
BASH	GNU bash, version 4.3.11(1)-release (x86_64-pc-linux-gnu)	The shell is a program that takes typed commands from the keyboard and passes them to the operating system or ROS to perform.
Robot Operating System (ROS)	"Indigo" Igloo	A software package used for Robotic software development. It is the interface between the OS and Baxter.
Python	Python 2.7.6	The Rethink Baxter Examples are written in Python.

Table 2. Versions of Software supporting Baxter

## BAXTER'S SYSTEM

Baxter the Robot can be controlled from the workstation and the various programs supplied by Rethink Robotics. Figure 3 shows how the workstation and Baxter are connected. Baxter has his own IP address.

The SDK interfaces with the Baxter Research Robot via [ROS](#) (Robot Operating System). Baxter provides a stand-alone ROS Master to which any development workstation can connect and control Baxter via the various [ROS APIs](#). The ROS and the APIs will be described later in this report.

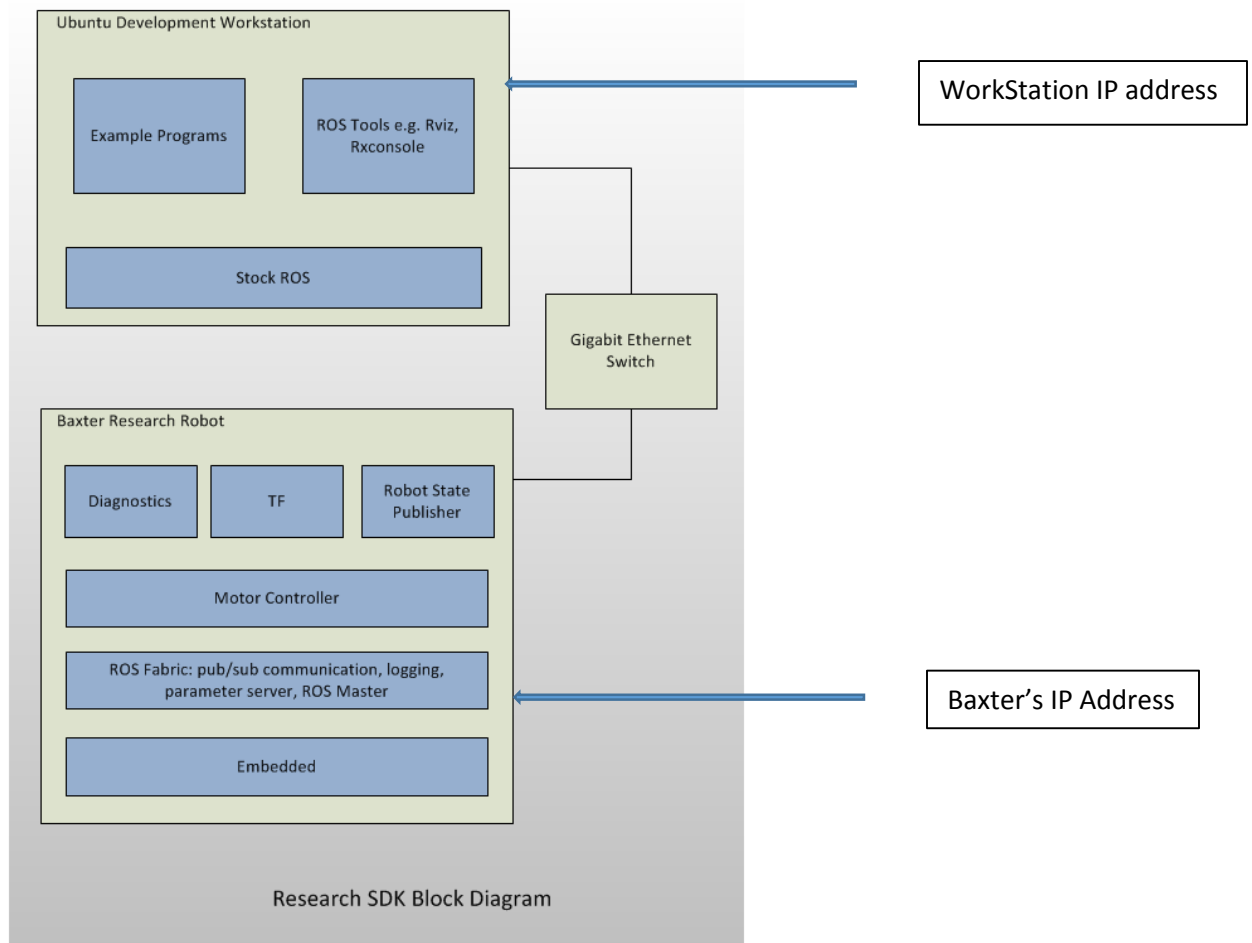


Figure 4 Overall Architecture of Baxter

[http://sdk.rethinkrobotics.com/wiki/Baxter\\_Research\\_Robot\\_Software\\_Developers\\_Kit\\_\(SDK\)](http://sdk.rethinkrobotics.com/wiki/Baxter_Research_Robot_Software_Developers_Kit_(SDK))

## BAXTER'S HARDWARE

This section describes Baxter's basic specifications. Appendix I gives more details. Some useful details are found in the various patents that have been filed or awarded to Baxter. For example,

<http://www.google.com/patents/US20130345872> **User interfaces for robot training**

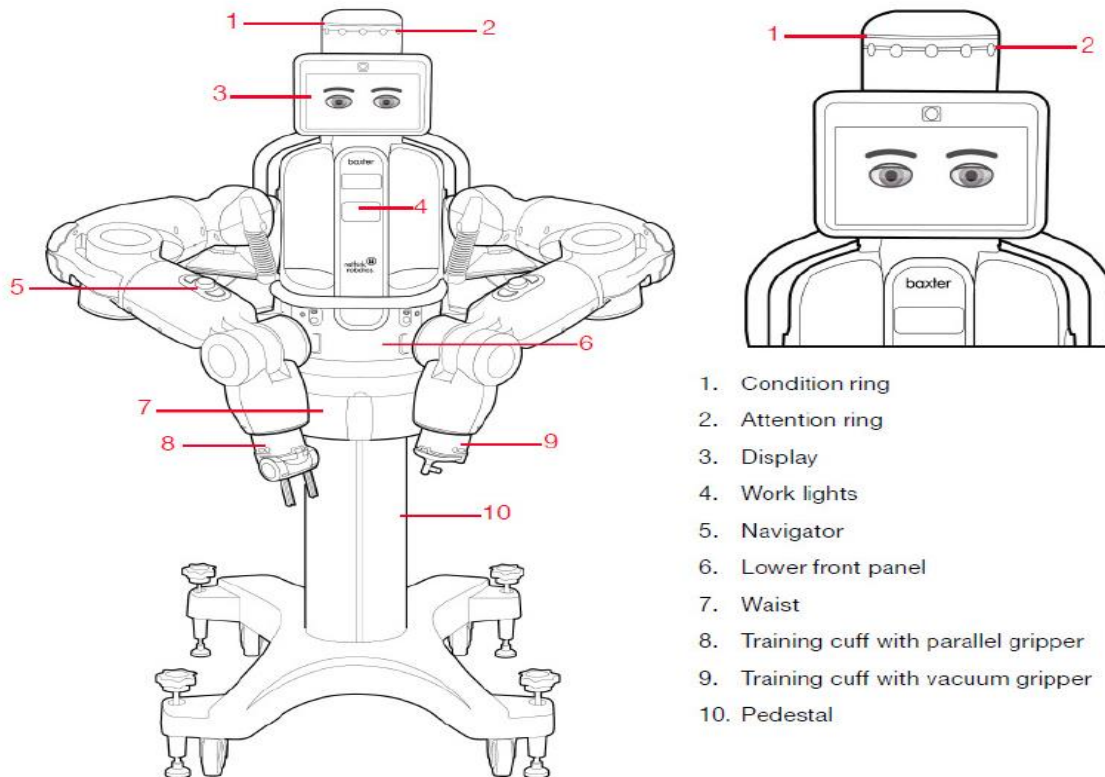


Figure 5 Baxter's Physical Structure

The *Pre-delivery Guide* and the Hardware Specifications describe Baxter's hardware:

[http://sdk.rethinkrobotics.com/mediawiki-1.22.2/images/9/94/BRR\\_pre-delivery\\_131120.pdf](http://sdk.rethinkrobotics.com/mediawiki-1.22.2/images/9/94/BRR_pre-delivery_131120.pdf)

[http://sdk.rethinkrobotics.com/wiki/Hardware\\_Specifications](http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications)

The Baxter Robot head display is a 1024 x 600 SVGA LCD screen capable of panning 180° and also nodding to acknowledge user input. Baxter's head has a pan joint and a single "Nod" action



for movement. There is one camera, one red LED ring, one green LED ring, and 12 individual yellow LEDs along with 12 sonar transducers. <http://sdk.rethinkrobotics.com/wiki/Head>

Some of the items in Figure 5 include:

1. Condition Ring – A green lighted ring will show when Baxter is ready to work (after the internal computer boots up)
2. Attention Ring – A ring of motion sensors that indicate the presence of a person or object
3. Display – The screen with resolution of 1024 x 600 pixels can be used for display of pictures or images or show text during Baxter’s operation.

In the **manufacturing version**, the display shows Baxter’s state when Baxter is being trained or is working. The operator uses the display to make selections via the navigator buttons in Figure 5 and Baxter uses the display to communicate information to the operator. This is explained in the User’s Guide for the manufacturing Baxter:

[http://mfg.rethinkrobotics.com/mfg-mediawiki-1.22.2/images/4/42/3.0.0\\_User\\_Guide.pdf](http://mfg.rethinkrobotics.com/mfg-mediawiki-1.22.2/images/4/42/3.0.0_User_Guide.pdf)

The following video shows a good example of training the manufacturing version of Baxter:

<https://www.youtube.com/watch?v=tWWKhp892Gk&feature=em-uploademail>

The training cuffs (8-9) in Figure 5 are explained in the section describing the ”zero-g mode” later. They are used extensively in the manufacturing version of Baxter but also have uses for the research Baxter. The cuffs are used basically to move Baxter’s arms without much resistance and also to open and close the electric gripper in certain operations.

Electrically, Baxter uses standard 120VAC power. The robot power bus and internal PC both have “universal” power supplies and support 90 - 264V AC (47 - 63Hz). The current is 6A at 120V AC or 720W maximum.

Baxter’s rear connections include the Power connector, Emergency Stop connection, USB port, Ethernet port, and Vacuum connection if a vacuum gripper is used.

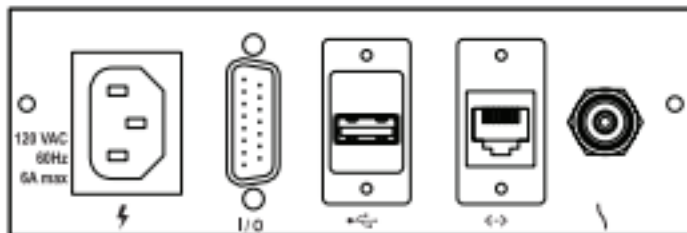


Figure 6 Baxter’s External Connections

Details of Baxter's reach and workspace are described in the workspace guidelines in the Baxter Setup Document: [http://sdk.rethinkrobotics.com/wiki/Workspace\\_Guidelines](http://sdk.rethinkrobotics.com/wiki/Workspace_Guidelines)

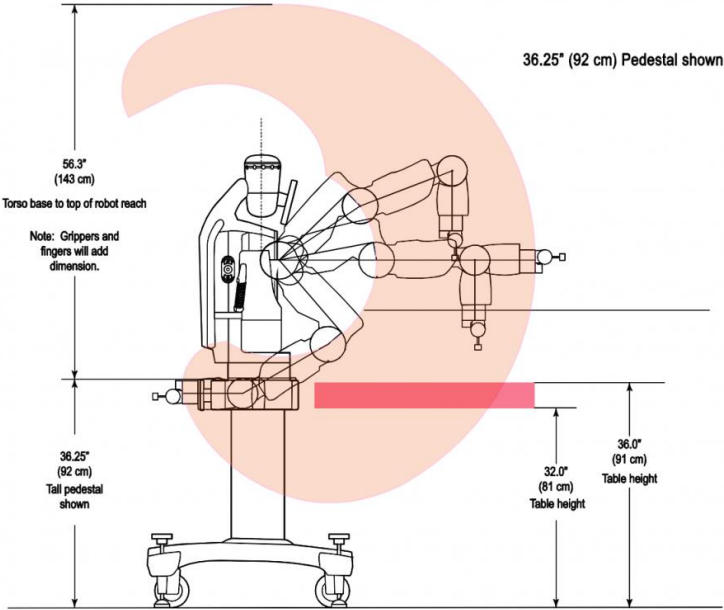


Figure 7 Side View of Baxter's Workspace

## Baxter's Internal Computer

Processor 3rd Gen Intel Core i7-3770 Processor (8MB, 3.4GHz) w/HD4000 Graphics

Memory 4GB, NON-ECC, 1600MHZ DDR3



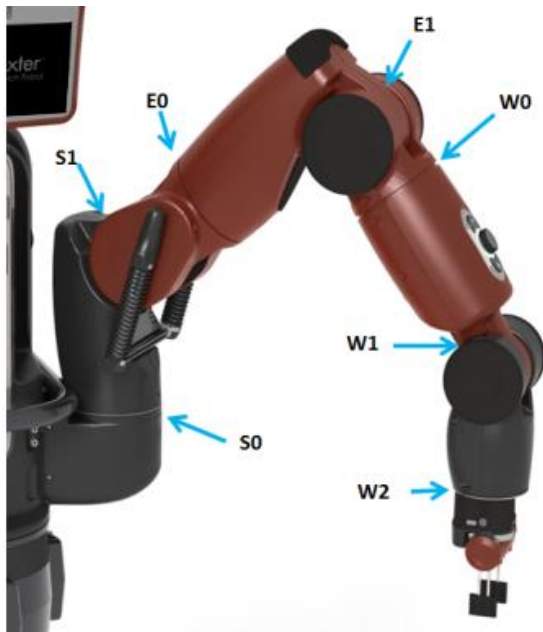
Figure 8 Baxter's Internal Circuits

Baxter's Field Service Menu (FSM) is used for setup during initial installation and other operations: <http://sdk.rethinkrobotics.com/wiki/FSM>

"The Field Service Menu (FSM) is a pre-boot configuration menu that allows the user to do advanced tasks such as check network interface configuration, change the robot computer's hostname, and run low-level hardware checks." This feature is normally not used by a Baxter User except for special operations. The FSM appears on Baxter's display when activated.

In Baxter's Lab, Baxter has his own IP address - <http://172.29.64.200:11311>. The details are presented in the document: <http://sdk.rethinkrobotics.com/wiki/Networking>

## Baxter's 7DOF Arms



- **S0** - Shoulder Roll
- **S1** - Shoulder Pitch
- **E0** - Elbow Roll
- **E1** - Elbow Pitch
- **W0** - Wrist Roll
- **W1** - Wrist Pitch
- **W2** - Wrist Roll

*Figure 9 Baxter's Arm and Joint Designations*

Baxter's 7 Degree of Freedom (DOF) compliant arms are described in several documents on the Rethink Robotics WEB site:

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

[http://sdk.rethinkrobotics.com/wiki/Hardware\\_Specifications](http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications)

The figure shows the arm and the end effector which is an electric gripper in this case. Rethink recommends: “Periodic recalibration will ensure that position and force sensors work well and the robot arms can move smoothly and accurately.”

[http://sdk.rethinkrobotics.com/wiki/How\\_often\\_does\\_baxter\\_need\\_to\\_be\\_calibrated](http://sdk.rethinkrobotics.com/wiki/How_often_does_baxter_need_to_be_calibrated)

For more advanced users familiar with ROS, the calibration procedure is described in the document:

[http://sdk.rethinkrobotics.com/wiki/Arm\\_Calibration](http://sdk.rethinkrobotics.com/wiki/Arm_Calibration)

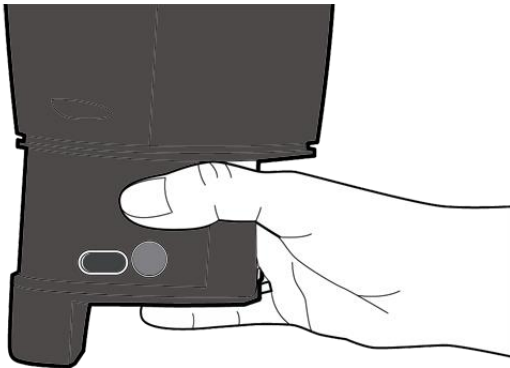
## Zero-G Mode

The zero-g mode allows control of Baxter’s arm without resistance from the arm’s motors. The mode is described in the document:

[http://sdk.rethinkrobotics.com/wiki/Zero-G\\_Mode](http://sdk.rethinkrobotics.com/wiki/Zero-G_Mode)

According to the description: “Zero-G mode can often be confused with the mode obtained by disabling the gravity compensation torques. By default, the gravity compensation torques will always be applied when the robot is enabled. In Zero-G mode, the controllers are disabled and so the arm can be freely moved across. In this case, the effect of gravity would be compensated by the gravity compensation model applying gravity compensation torques across the joints, there would be no torques from the controllers since they would not be active, and so the arm can be moved freely around, hence the name.”

“The Zero-G mode can be enabled by grasping the cuff over its groove as below



*Figure 10 Baxter's Cuff to Enable Zero-G mode*

## References for the Arms and the Series Elastic Actuators:

The arms have a compliance which allows close and safe human interaction with Baxter. This is accomplished through a series of elastic actuators incorporated into all 14 arm joints.

### Part Manipulation using Sensing and Force Control – Alex Goodwin, Rethink Robotics

“Alex described Rethink Robotics flagship product, Baxter, as a robot falling into a new category of robotics defined by industry requirements for dexterity. As such, the Baxter platform offers

direct manipulation as well as integral vision and force sensing of two highly dexterous 7 DOF arms. This new technology is designed to work alongside people performing human-scale tasks at human cadence, eliminating the need for safety cages. This is accomplished through a series of elastic actuators incorporated into all 14 arm joints.” **NISTIR 7940**

**Dexterous Manipulation for Manufacturing Applications Workshop** June 2013

<http://www.nist.gov/el/isd/upload/NIST-IR-7940.pdf>

**A.I. Technical Report No. 1524 January, 1995**

**Series Elastic Actuators, Matthew M. Williamson**

<http://dspace.mit.edu/bitstream/handle/1721.1/6776/AITR-1524.pdf?sequence=2>

**Baxter’s Electric Gripper**

The UHCL Baxter has an electric gripper. Installation is described in the document:

[http://sdk.rethinkrobotics.com/wiki/Electric\\_Gripper\\_Installation](http://sdk.rethinkrobotics.com/wiki/Electric_Gripper_Installation)



*Figure 11 Baxter's Electric Gripper*

The gripper has two “fingers” with removable inserts to allow different configurations of the gripping surface. According to the specifications”

Positional Accuracy +/- 5 mm

Max Payload (including end-effector) 5 lb / 2.2 kg

Gripping Torque (max) 10 lb / 4.4 kg

Gripper maximum opening width can be adjusted by repositioning the “fingers” as described in the document:

[http://sdk.rethinkrobotics.com/wiki/Electric\\_Gripper\\_Installation](http://sdk.rethinkrobotics.com/wiki/Electric_Gripper_Installation)

“Each hand has a 3-axis accelerometer located inside the cuff, in the same plane as the gripper electrical connection header. The positive z-axis points back 'up' the arm (towards the previous wrist joint, w0). The positive x-axis points towards the camera, and the y-axis points towards the cuff buttons, using standard [Right-Hand-Rule](#) notation” as described in the reference:

<https://github.com/RethinkRobotics/sdk-docs/wiki/API-Reference>

This reference is for advanced users as it details the API and ROS topics for the Baxter hardware.

## Baxter's Camera and Sensors

Baxter has a camera in each cuff as well as an Infrared (IR) sensor to measure distance.



*Figure 12 Baxter's camera and IR Sensor in the Cuff*

The figure shows the end of the cuff with the cuff camera and the IR ranging sensor.

The Camera Specifications are as follows:

Max Resolution	1280 x 800 pixels
Effective Resolution	640 x 400 pixels
Frame Rate	30 frames per second
Focal Length	1.2mm

Infrared (IR) Range Sensors in each cuff have a range of 1.5 – 15 in / 4 – 40 cm

The documents describe the cuff and the sensors in more detail:

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

[http://sdk.rethinkrobotics.com/wiki/API\\_Reference#Sensors](http://sdk.rethinkrobotics.com/wiki/API_Reference#Sensors)



Experiments in Baxter's lab made several things clear:

1. The accuracy of the distance measurement using the IR sensor varied depending on the surface below the sensor.
2. Note from the figure of the cuff camera that the center of the gripper is offset in both the x and y direction if we consider the z direction pointing up the arm when the gripper is in line with the first joint. Thus, if the camera is used for positioning the gripper over an object to be picked up, the offsets must be considered.



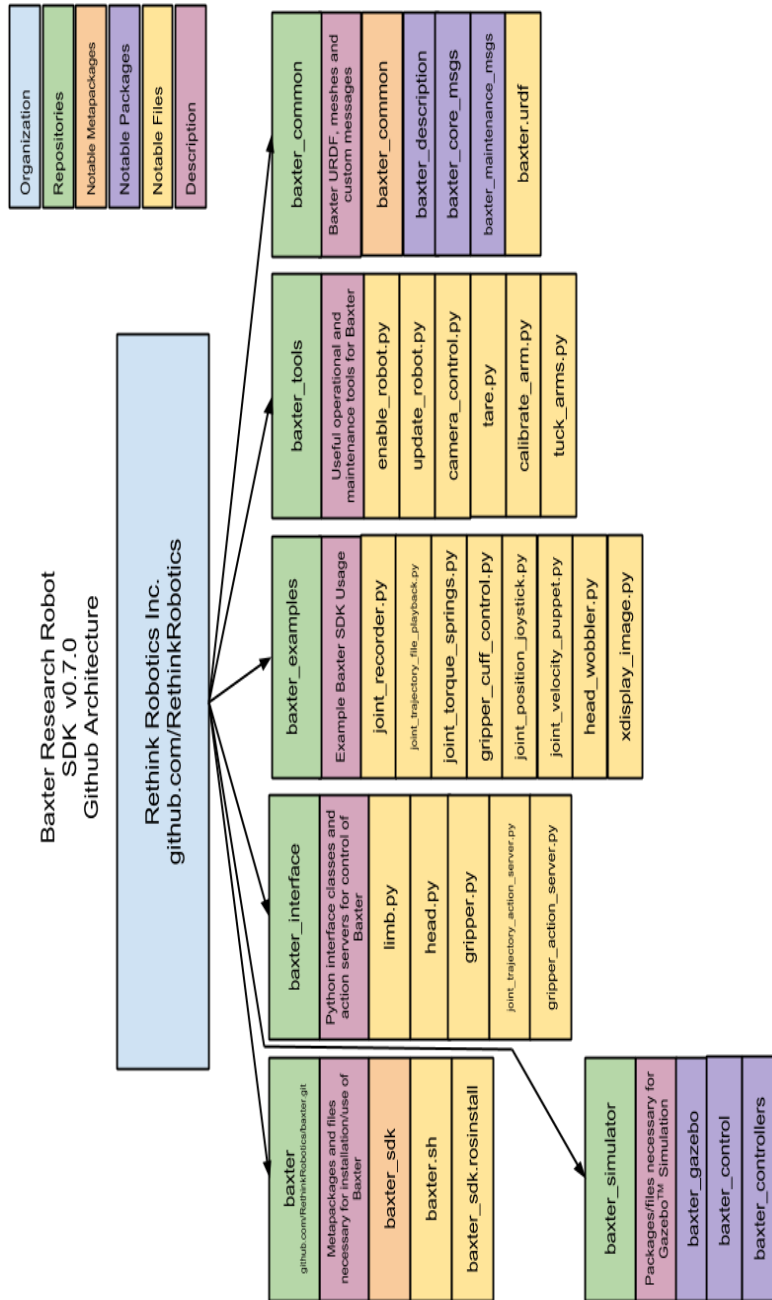
This article is a simple discussion of IR and sonar sensors:

[http://www.societyofrobots.com/member\\_tutorials/node/71](http://www.societyofrobots.com/member_tutorials/node/71)

Generally, a black object that is not reflective will absorb some of the IR and the distance measurement might be in error. A white table top, for example, reflects the IR very well and Baxter's IR sensor calculated the distance from the table accurately and repeatedly.

## SDK SOFTWARE FOR BAXTER

The next 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. These will be described in detail in another report *Baxter User's Guide* which will describe the use of Baxter and its standard applications called "Rethink Baxter Examples" provided by Rethink Robotics.



## BAXTER SIMULATORS

### MoveIt

The Rethink document presents a detailed tutorial about the use of MoveIt.

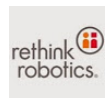
[http://sdk.rethinkrobotics.com/wiki/MoveIt\\_Tutorial](http://sdk.rethinkrobotics.com/wiki/MoveIt_Tutorial)

“MoveIt! motion planning framework provides capabilities including Kinematics (IK, FK, Jacobian), Motion Planning (OMPL, SBPL, CHOMP) integrated as MoveIt! plugins, Environment Representation (robot representation, environment representation, collision checking, constraint evaluation), execution using move\_groups, benchmarking, warehouse database for storage (scenes, robot states, motion plans), a C++/Python API and more!”

In short, MoveIt allows path planning with obstacles in Baxter’s path and then execution of the path on the “real” Baxter.

A very useful video from Rethink explains the use of MoveIt.

Baxter Research Robot MoveIt! Tutorial



**Rethink Robotics**

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

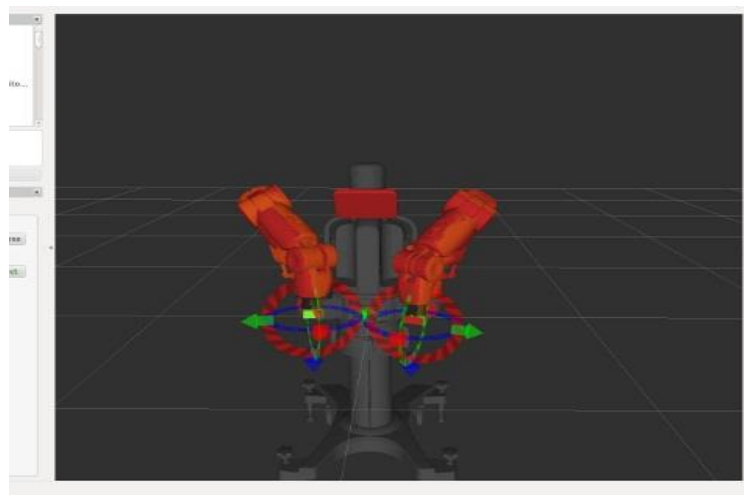


Figure 14 MoveIt Simulation of Baxter

To use MoveIt, the Rviz (ROS visualization) window will then open showing Baxter with interactive markers. The markers are used to move Baxter’s arms in the simulation.

“[rviz \(ROS visualization\)](#) is a 3D visualizer for displaying sensor data and state information from ROS. Using rviz, you can visualize Baxter's current configuration on a virtual model of the robot. You can also display live representations of sensor values coming over ROS Topics including camera data, infrared distance measurements, sonar data, and more.”

## Baxter Simulator Gazebo

The site [http://sdk.rethinkrobotics.com/wiki/Baxter\\_Simulator](http://sdk.rethinkrobotics.com/wiki/Baxter_Simulator) describes the Baxter Simulator. It is used to model Baxter and environments. Rethink defines Baxter's characteristics in the form of a URDF (Unified Robot Description Format). Baxter automatically builds his URDF on boot-up. It is accessible by ROS utilities, Gazebo and other simulators.

"Gazebo is a multi-robot simulator in a 3-dimensional world. It comes with advanced plugin interfaces that can be used to simulate the sensor feedback and plausible interactions between objects. There are standard models available within gazebo that can be used with custom models."



Figure 15 Baxter Simulator

## V-REP (Virtual Robot Experimentation Platform)



“The video shows the Baxter robot in the V-REP robot simulator ( <http://www.coppeliarobotics.com> ). It illustrates several of V-REP's features: simple scene composition, physics, motion planning, inverse kinematics, etc. The video shows the Baxter robot doing some object manipulation. The simulation uses V-REP's built-in motion planning and inverse kinematics algorithms. The Baxter CAD data is courtesy of Rethink Robotics.”

<https://www.youtube.com/watch?v=PHuQZDp8kt8>

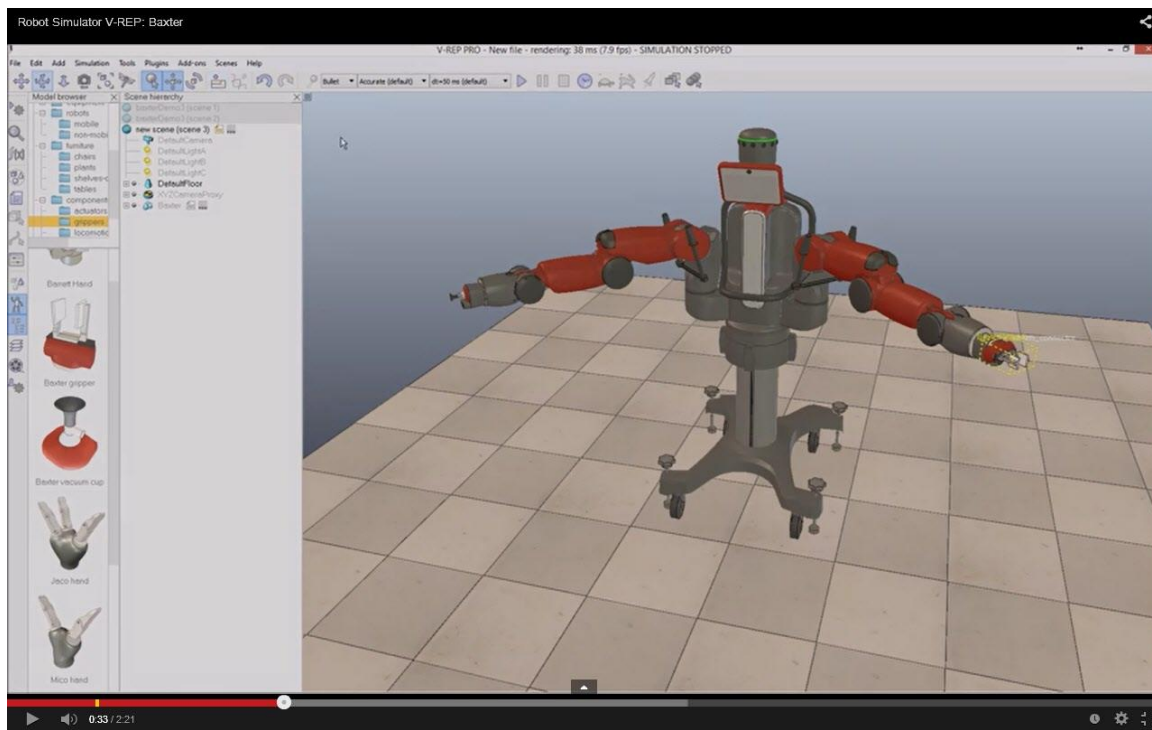


Figure 16 VREP Simulation of Baxter

The User's Manual is available at this site: <http://www.coppeliarobotics.com/helpFiles/index.html>

## ROBOT OPERATING SYSTEM (ROS)

ROS is sometimes called a “meta operating system” because it performs many of the functions of an operating system. One of its main purpose is to provide communication between the user, the Ubuntu OS and of course Baxter. As with any operating system, the benefit of ROS is the hardware abstraction and low-level control of Baxter without the Baxter user knowing all of the details of the robot. For example, to move Baxter’s arms, a command to ROS is issued and ROS communicates with the scripts<sup>1</sup> in Python written by the Rethink designers to cause Baxter to respond as commanded. Baxter is defined to ROS by the designers at Rethink Robotics in various ways describe later.

For a little history:

[http://en.wikipedia.org/wiki/Robot\\_Operating\\_System](http://en.wikipedia.org/wiki/Robot_Operating_System)

“ROS was originally developed in 2007 under the name *switchyard* by the [Stanford Artificial Intelligence Laboratory](#) in support of the Stanford AI Robot STAIR project. From 2008 until 2013, development was performed primarily at [Willow Garage](#), a robotics research institute/incubator. During that time, researchers at more than twenty institutions collaborated with Willow Garage engineers in a federated development model.

In February 2013, ROS stewardship transitioned to the Open Source Robotics Foundation. In August 2013, a blog posting<sup>[1]</sup> announced that Willow Garage would be absorbed by another company started by its founder, Suitable Technologies. The support responsibilities for the PR2 created by Willow Garage were also subsequently taken over by Clearpath Robotics”.

The link to Clearpath Robotics is <http://www.clearpathrobotics.com/>

Our Baxter now (2015) uses the Indigo release of ROS:

ROS Indigo Igloo is the eighth [ROS distribution release](#) and was released July 22nd, 2014. Quote from: <http://wiki.ros.org/indigo>

---

<sup>1</sup> The Python scripts in turn call various software elements including the Baxter APIs (application programming interfaces) that cause the actual motion of Baxter’s arms.

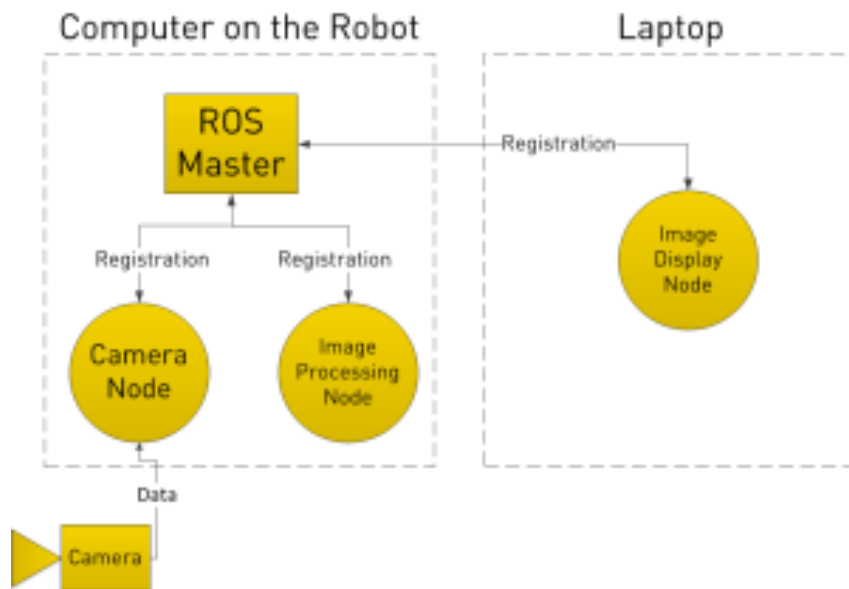
## ROS Terms

Before a beginner even opens a web tutorial or book or sees a ROS video, it is helpful to learn a few terms that pertain to ROS. These terms describe the main components of a ROS system.

Item	Type	Comment
Repositories	A <b>software repository</b> is a storage location from which <a href="#">software packages</a> may be retrieved and installed on a computer.	<a href="http://en.wikipedia.org/wiki/Software_repository">http://en.wikipedia.org/wiki/Software_repository</a>  GitHub is used to download the ROS packages used by the Baxter system: <a href="http://sdk.rethinkrobotics.com/wiki/Workstation_Setup">http://sdk.rethinkrobotics.com/wiki/Workstation_Setup</a>
Packages	Contains files to allow execution of ROS programs	A package typically contains source files and executable scripts that can be BASH, Python, or other code.
Manifest	Information about a package	The manifest defines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other packages.
Services	Allows communication between nodes.	Used by nodes to communicate with other nodes and request a response.
Nodes	Processes that execute commands.	Executable code written in Python or C++ usually. Python nodes use the client library <i>rospy</i>
ROS Master	Registers the name and location of each node.	Allows nodes to communicate. Nodes can be in different computers.
Messages	Data sent between nodes.	Messages are “published” by a node and “subscribed to” by another node.
Topic	Name of a message.	For example, Baxter’s cameras “publish” the image they receive as a topic with a name that indicates it is a camera image.
Bags	Data storage for messages.	Used to save and playback data such as sensor data.


Table 3 ROS Terms

Figure 17 Example ROS nodes from Clearpath Robotics



## ROS Tutorials and Books

A large number of tutorials are available at the ROS.ORG site: <http://wiki.ros.org/ROS/Tutorials>

 The tutorials at this site and many other tutorials should probably be called “ROS tutorials for those familiar with Ubuntu, Python or C++, and ROS itself”. Check to see that the references or books cover your version of the software.

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

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

## ROS Workspace

The files for ROS and Rethink Baxter examples are downloaded into the ROS workspace when the system is created by downloading the software from the GitHub repositories. The instructions from Rethink Robotics are located at the web site:

[http://sdk.rethinkrobotics.com/wiki/Workstation\\_Setup](http://sdk.rethinkrobotics.com/wiki/Workstation_Setup)

The setup procedure is not discussed here as it is assumed that this Introduction to Baxter will be used by those whose system is installed.




The command that causes communication between the workstation and Baxter is invoking the Baxter shell from the terminal window with a command such as: `./baxter.sh`

The result would indicate Baxter's IP address, the user, and that the current directory is `ros_ws` in the example from Baxter's lab at UHCL:

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

Baxter is now ready to execute user commands.

 The examples described in this report are from the UHCL Baxter lab. Any other setup of the Baxter software possibly would use different names for the directories, different IP addresses and certainly different user names for logging in to the system. These details must be determined by the system administrator who setup the system.

Details about linking the workstation and Baxter are describe in this Rethink document:

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

It is intended for those who understand networks and ROS Naming Conventions. For advanced users, the ROS tutorial might be helpful: <http://wiki.ros.org/ROS/NetworkSetup>

## UBUNTU AND BASH

Borrowing from Wikipedia, the free encyclopedia:

**Ubuntu** (/uːˈbuːntuː/ *oo-**BOON**-too*) is a [Debian](#)-based [Linux operating system](#), with [Unity](#) as its default [desktop environment](#) ([GNOME](#) was the previous [desktop environment](#)). It is based on [free software](#) and named after the Southern African philosophy of [ubuntu](#) (literally, "human-ness"), which often is translated as "humanity towards others" or "the belief in a universal bond of sharing that connects all humanity". [http://en.wikipedia.org/wiki/Ubuntu\\_\(operating\\_system\)](http://en.wikipedia.org/wiki/Ubuntu_(operating_system))

There are so many books, articles, videos and other references to Ubuntu or Linux or Unix as operating systems that the list would be longer than this report. Here is a downloadable, reasonably readable reference with the advantage it is only 143 pages long!




# Getting Started with Ubuntu 14.04



Figure 18 Getting Started with Ubuntu 14.04

[http://files.ubuntu-manual.org/manuals/getting-started-with-ubuntu/14.04/en\\_US/screen/Getting%20Started%20with%20Ubuntu%2014.04.pdf](http://files.ubuntu-manual.org/manuals/getting-started-with-ubuntu/14.04/en_US/screen/Getting%20Started%20with%20Ubuntu%2014.04.pdf)

 A beginners question is surely “How much Ubuntu do I need to use Baxter?” Good question! A system administrator needs knowledge of user accounts and permissions. Also, some software downloads require administrator privilege. Fortunately, a user of Baxter can forgo the extensive study needed to be proficient in OS techniques. This report concentrates on users, NOT administrators.

Of course, any user would need to create directories if not created by the Administrator, use editors to create programs and save the programs, list files in a directory, look up information on

the web, print text, and generally use the workstation efficiently. The Ubuntu system has two interfaces for the user, one graphic called the **Unity Interface** and one command line interface called the **Terminal Interface**.

## Unity Interface

The most convenient interface for a casual user is the Unity desktop. It is a graphical user (GUI) interface that will not be strange to Windows users or especially MAC OSX users.

This article on the WEB is useful for beginners:

*How to Master Ubuntu's Unity Desktop: 8 Things You Need to Know*

<http://www.howtogeek.com/113330/how-to-master-ubuntus-unity-desktop-8-things-you-need-to-know/>

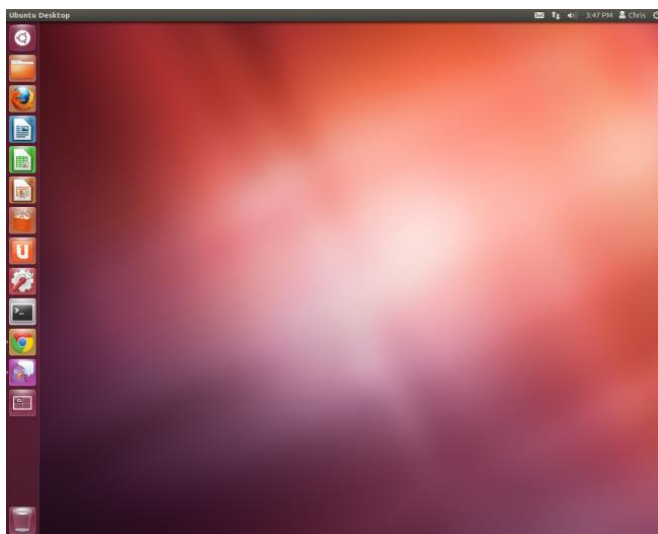


Figure 19 Unity Desktop

The Unity user interface consists of several components:

- **Top menu bar** – a multipurpose top bar, saving vertical space, and containing: (1) the menu bar of the currently active application, (2) the capture bar of the main window of the currently active application including the maximize, minimize and exit buttons, (3) the global system pulldown menu including the global system settings, logout, shut down and similar basic controls, and (4) the small icons that indicate the time, sound control, wi-fi signal strength, the user and perhaps other information.
- **Launcher** – a [dock](#) on the left side vertically that also serves as a window switcher. Click on an icon to execute the application. Frequently used applications should be pinned to the Launcher by moving their icon from the desktop to the Launcher bar. The “Trash” icon at the bottom of the Launcher serves to hold deleted files and documents.

In Figure 19, the first five applications from top to bottom of the launcher are as follows:

**Dash** – an overlay that allows the user to search quickly for information both locally (installed applications, recent files, bookmarks, etc.) and remotely (Twitter, Google Docs, etc.) and displays previews of results.

**File Manager** – lists and allows navigation to various directories including the File System, Documents, Pictures, etc. The file manager shows the root (/) folder, the home folder for the user and the home folder for the system. Here is a description of some of the root directories for Ubuntu 10.04 – an older version than we are using:

<https://help.ubuntu.com/10.04/installation-guide/amd64/directory-tree.html>



Be careful with documentation for any software module because all of the software described in this report is subject to updating by the organizations or individuals that maintain the software.

**Mozilla Firefox** – the WEB browser useful for searches using Google and bookmarking relevant documents for Baxter. One of the first sites to bookmark is the “wiki/Main\_Page” from Rethink Robotics:

[http://sdk.rethinkrobotics.com/wiki/Main\\_Page](http://sdk.rethinkrobotics.com/wiki/Main_Page)

This is the research Baxter’s main information page which is invaluable for setting up Baxter and using the Baxter Examples provided by Rethink.

**LibreOffice and LibreOfficeCalc** - word processing and spreadsheet programs compatible with Word and Excel.

Explore the other icons used for software downloads, system settings and other applications.

Many normal operations can be performed using the Unity GUI desktop. Folders can be created in any directory by Right Clicking in the directory opened in the File Manager and using the drop down menu and selecting Create New Folder, for example. There is a search option in the File manager also. If an external USB device is connected such as a memory stick, its icon will be displayed in the list of files and on the Launcher dock.

## Terminal Interface

For Baxter users, the terminal interface is the most important interface on the desktop. If it is not installed, activate the Dash, search for “terminal” and drag the icon to the desktop and then to the launcher.

Here is a reference that may be of questionable use to a beginner

<https://help.ubuntu.com/community/UsingTheTerminal>

There are a number of basic tutorial videos on YouTube. For example,

*Ubuntu 12.04 Terminal Tutorial #1* by Nicholas Lindsay:

[http://www.youtube.com/watch?v=Uabfu\\_RJLyg](http://www.youtube.com/watch?v=Uabfu_RJLyg) On that page are many other tutorials.

Some examples that compare Unity GUI commands with Terminal commands are shown in Appendix II.

The **terminal commands** are used as follows by Baxter users depending on their familiarity with Ubuntu and the terminal shell:

1. Perform all the usual operations expected of an operating system such as creating directories, files, documents, etc.
2. Create programs (scripts) that will command Baxter. There are text editors such as gedit, emacs or vi for this purpose. These editors do not insert any fancy formatting commands in the text.
3. Allow communication between Ubuntu, ROS, and Baxter as needed. For example, a command to enable Baxter would be typed on the terminal as

```
$ rosrun baxter_tools enable_robot.py -e
```

The \$ is the command prompt. When the command is executed, ROS executes the Python script `enable_robot.py` from the package `baxter_tools`. The Baxter Python programs and their connection to Baxter are discussed later.

4. Allow users to download software for Baxter or ROS that does not require administrative privileges. System administrators normally use the terminal commands for setting up and maintaining the system by installing and updating software.



This site is a good tutorial for those beginning to use the Ubuntu terminal commands. Note that different versions of the OS may have slightly different command formats.

“A beginners guide to the Unix and Linux operating system. Eight simple tutorials which cover the basics of UNIX / Linux commands”. <http://www.ee.surrey.ac.uk/Teaching/Unix/>

## What is BASH ?

Bash is the shell, or command language interpreter, for the GNU operating system. The name is an acronym for the ‘Bourne-Again SHell’, a pun on Stephen Bourne, the author of the direct ancestor of the current Unix shell `sh`, which appeared in the Seventh Edition Bell Labs Research version of Unix.

<http://www.gnu.org/software/bash/manual/bash.html>

Basically, a shell is simply a macro processor that executes commands. The term macro processor means the text and symbols of a command are expanded to create larger expressions.

A Unix shell is both a command interpreter and a programming language. As a command interpreter, the shell provides the user interface to the rich set of GNU utilities<sup>2</sup>. The programming language features allow these utilities to be combined. Files containing commands can be created, and become commands themselves. These new commands have the same status as system commands in directories such as `/bin`, allowing users or groups to establish custom environments to automate their common tasks.

Shells may be used interactively or non-interactively. In interactive mode, they accept input typed from the keyboard. When executing non-interactively, shells execute commands read from a file.

For advanced Baxter users, Baxter can be controlled by “shell scripts” which are simply programs that execute ROS commands or Rethink Baxter Examples when invoked. Fortunately, for those who are content running Baxter commands one by one from the terminal window’s command line, there is little need to learn Bash scripting. The Rethink Baxter Examples are discussed in the separate report *Baxter User’s Guide*.

The shell has a number of ways to obtain help for a command including the commands:

*info, man, -h, --help*

According to the *The Linux Information Project* website: <http://www.linfo.org/man.html>

“The man pages are a user manual that is by default built into most Linux distributions (i.e., versions) and most other Unix-like operating systems during installation. They provide extensive documentation about commands and other aspects of the system, including configuration files, system calls, library routines and the kernel (i.e., the core of the operating system). A configuration file is a type of simple database that contains data that tells a program or operating system how to behave. A system call is a request made via a software interrupt (i.e., a signal to the kernel initiated by software) by an active process for a service performed by the kernel. A library routine is a subprogram that is used by programmers to simplify the development of software”.

---

<sup>2</sup> <https://www.gnu.org/>



“The descriptions are rather terse, and they can seem somewhat cryptic to new users. However, users typically find them to be increasingly useful as they become more familiar with them and gain experience in the use of Unix-like operating systems”.<sup>3</sup>

## Bash confusion

As expected, there are many tutorials that cover the BASH shell. Here is one with a few of the comments from the site. For example, *Bash by Example*

<http://www.ibm.com/developerworks/library/l-bash/>



“Learning bash the wrong way can be a very confusing process. Many newbies type "man bash" to view the bash man page, only to be confronted with a very terse and technical description of shell functionality. Others type "info bash" (to view the GNU info documentation), causing either the man page to be redisplayed, or (if they are lucky) only slightly more friendly info documentation to appear.

While this may be somewhat disappointing to novices, the standard bash documentation can't be all things to all people, and caters towards those already familiar with shell programming in general. There's definitely a lot of excellent technical information in the man page, but its helpfulness to beginners is limited”.

This warning to new users should not discourage them from working with the shell and its commands. Don't expect instant success if you are accustomed only to the GUI interfaces of Operating Systems.

---

<sup>3</sup> Sorry to be flippant but I am reminded of the advertisements for “Weight Loss Pills”. They work when combined with a sensible regime of diet and exercise! If I followed their directions why would I need the pills.

## CONCLUSIONS

If you have managed to review this *Introduction to Baxter*, you could benefit from reviewing some of the documentation on the Rethink Robotics wiki site.

The “Learning” page appears as follows <http://sdk.rethinkrobotics.com/wiki/Learning>

The screenshot shows the 'Learning' page on the Rethink Robotics wiki. The page has a dark header with the 'baxter' logo and a search bar. Below the header is a red banner with the word 'Learning' and buttons for 'Discussion', 'Page', and 'More'. The main content is a grid of four categories, each with an icon and a list of sub-links. The 'Advanced Understanding' category is highlighted with a tooltip.

Foundations	Advanced Understanding	Tools	Configuration/Admin
<b>SDK Foundation</b>	<b>Further Understanding &amp; Advanced Concepts</b>	<b>External Tools</b>	<b>Setup, System Admin &amp; Maintenance</b>
First Steps Writing Programs Robot Foundations Resources	Conceptual Overview Baxter Platform Advanced Concepts	Operational Tools External Tools Supported ROS Tools Resources ROS Tools Links	Robot Configuration Software Updates Maintenance/Debug Tools Resources System Info

Figure 20 Rethink Robotics Learning Page

Each link on the page leads to expanded explanations of any item.

The “Foundations” page discussed Baxter applications in some detail. Many of the items presented there will be explained in our *Baxter User’s Guide*.

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

For those new to Baxter, the Rethink documentation should be used “many times” as each time new information becomes apparent and previous descriptions become clearer. In Baxter’s Lab, a combination of reviewing the documentation and using Baxter has led to a good understanding of Baxter’s capabilities.



## APPENDIX I Baxter Specifications

### 1.1 Specification:

#### PHYSICAL SPECIFICATIONS

<b>Robot Height</b>	3' 1" without pedestal 5' 10" – 6' 3" with adjustable pedestal (optional)
<b>Arm Length to End Effector Plate</b>	41"
<b>Torso Mounting Plate Diameter</b>	13.3" (for mounting on table)
<b>Body weight</b>	165 lbs. without pedestal 306 lbs. with pedestal (optional)
<b>Degrees of Freedom</b>	14 (7 per arm)
<b>Pedestal Footprint</b>	36" × 32"
<b>Max Pay Load (Including End Effector)</b>	5 lb / 2.2 kg
<b>Gripping Torque (Max)</b>	10 lb / 4.4 kg

## COMPUTER & SENSOR SPECIFICATIONS

<b>Processor</b>	3rd Gen Intel Core i7-3770 Processor (8MB, 3.4GHz) w/HD4000 Graphics
<b>Memory</b>	4GB, NON-ECC, 1600MHZ DDR3
<b>Storage</b>	128GB Solid State Drive
<b>Camera Max Resolution</b>	1280 x 800 pixels
<b>Camera Effective Resolution</b>	640 x 400 pixels
<b>Camera Frame Rate</b>	30 frames per second
<b>Camera Focal Length</b>	1.2 mm
<b>Screen Resolution</b>	1024 x 600 pixels
<b>Infrared Sensor Range</b>	1.5 – 15 in / 4 – 40 cm

## ELECTRICAL SPECIFICATIONS

<b>Supply Voltage</b>	120 volts alternating current
<b>Rated Current</b>	6 amps
<b>Battery Operation</b>	DC-to-120V AC Inverter (Note: the Baxter robot has an internal PC, which cannot be powered directly off of 24V DC)
<b>Interface</b>	Standard 120VAC power. Robot power bus and internal PC both have “universal” power supplies and support 90 – 264V AC (47 – 63Hz)
<b>Max Consumption</b>	6A at 120V AC, 720W max per unit
<b>Maximum Efficiency</b>	87% to 92%
<b>Power Supply</b>	Uses medical-grade DC switching power supply for robot power bus
<b>Tolerance to Sags</b>	Sags tolerated to 90V. Sustained interruption will require manual power-up
<b>Voltage Flicker</b>	Holdup time 20mS
<b>Voltage Unbalance</b>	Single phase operation only

## WORKSTATION REQUIREMENTS

Baxter Research Robot requires a workstation (not included) that meets the following technical specifications:

<b>Software</b>	Ubuntu 10.4 LTS and ROS Electric - Upgraded to 12.04 LTS and ROS Indigo- 2015
<b>Processor</b>	Intel i5 or above
<b>Memory</b>	4GB or more
<b>Free Disk Space</b>	2GB or more
<b>Connectivity</b>	Ethernet port

## APPENDIX II Unity and Terminal Commands

Comparison of Unity GUI commands and Terminal commands			
UNITY EXAMPLES		TERMINAL EXAMPLES	
<b>GENERAL HELP</b>			
Dash Help ? F1 –	Desktop Guide  Works with many applications	cmd <sup>1</sup> --help help cmd man cmd whatis cmd info cmd	cp –help help pwd man echo whatis grep info grep
<b>FILES DIRECTORIES</b>			
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
<b>EDITOR</b>			
LibreOffice	Save as txt if a program- change suffix	gedit file	Move up to menu bar and File:SaveAs
<b>SYSTEM</b>			
Sysinfo in Dash SystemSettings-Launcher or on Right of menu bar	(Installed) -Hardware Appearance, HW, Network...,details, User Accounts	lspci -v	More information than you need
<b>NETWORK</b>			
SystemSettings Network	172.29.64.201	ifconfig	

<sup>1</sup> Here cmd means any valid command such as ls