





# baxter

## Research Robot

### Technical Specification Datasheet & Hardware Architecture Overview

The Baxter Research Robot is a compelling addition to the world of Research and Education. With the same industry-tested hardware of Rethink Robotics' flagship robot Baxter that is revolutionizing the manufacturing world, the Baxter Research Robot is providing a safe, affordable, robust platform for universities and labs to utilize in exciting ways. With market-leading value, a unique inherent safety system and the real-world relevance of the Baxter manufacturing solution, it is quickly becoming a must-have tool for leading institutions around the globe.

Baxter consists of two, 7-degree-of-freedom arms with Series Elastic Actuators at each joint, incorporating full position and force sensing. Three integrated cameras, along with sonar, accelerometers and range-finding sensors. Baxter's intuitive Zero Force Gravity Compensation mode allows users to effortlessly move each 7-degree-of-freedom arm and teach the robot positions and trajectories without the need to program them mathematically, enabling collaborative human-robot co-work and advanced Human Robot Interaction.

The Baxter Research Robot allows direct programming access to the system via a standard, open-source ROS API interface. Users can run custom programs from a connected development workstation, or locally through access to the on-board CPU. It is entirely safe to operate around humans without the need for safety cages or other guarding equipment, and comes with a suite of example programs, demonstration videos, online documentation and a robust user community to help new users get started quickly in developing new applications.

Further technical details can be found at: <http://sdk.rethinkrobotics.com>

Open-Source SDK Code can be found at: <https://github.com/RethinkRobotics>

API Documentation can be found at: <http://api.rethinkrobotics.com/>

# Hardware Architecture Overview:

## Arms:

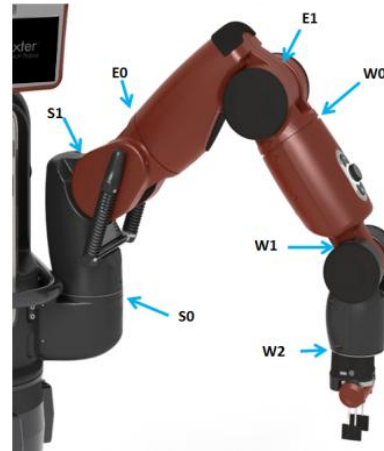
- ▲ Baxter has two seven degree-of-freedom (DOF) arms. Seven DOF arms are desirable as they provide a kinematic redundancy greatly improving manipulability and safety.
- ▲ Each joint contains a Series Elastic Actuator (SEA), this actuation technique is key to making Baxter safe.
  - ▲ Series Elastic Actuators differ from traditional actuation techniques in that we introduce a spring between the motor/gearing elements and the output of the actuator. This results in:
    - ▲ gains in stable, low noise force control
    - ▲ protection against shock loads
  - ▲ Inherent safety is a characteristic of SEAs. The springs in these actuators are deformable by human level inputs. This deflection is an inherent safety mechanism.
  - ▲ By introducing the spring element we are now able to measure torque output from the actuator. This enables such things as torque control, impedance control, and more.
  - ▲ Rethink Robotics', Matthew Williamson's Paper on the SEA architecture can be found at: [http://groups.csail.mit.edu/lbr/hrg/1995/mattw\\_ms\\_thesis.pdf](http://groups.csail.mit.edu/lbr/hrg/1995/mattw_ms_thesis.pdf)



## **A Joint Naming Convention:** Baxter's Joints are named using the following convention.

### **A** From the Base, along the arm to the End-Effector:

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

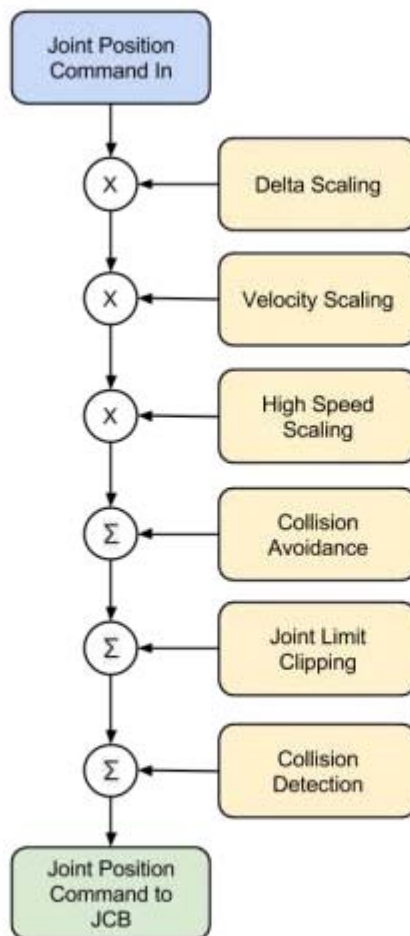


## **A Joint Control**

- A** Fundamental to the operation of Baxter via the SDK is a familiarity with the available joint control modes. The joint control modes define the 'modes' in which we can control Baxter's limbs in joint space. These are provided via a ROS message API.
- A** When a joint position command is published from the development PC, the 'realtime\_loop' process which represents a motor control plugin subscribes to this message. This message is then parsed, and represented in memory based on control mode.
- A** Depending on the control mode, modifications are made to the input commands. These modifications are typically due to the safety controllers (e.g. arm-to-arm collision avoidance, collision detection, etc.)
- A** A control rate timeout is also enforced at this motor controller layer. This states that if a new 'JointCommand' message is not received within the specified timeout (0.2 seconds, or 5Hz), the robot will 'Timeout'. When the robot 'Times out', the current control mode is exited, reverting back to position control mode where the robot will command (hold) it's current joint angles. The reason for this is safety. For example, if controlling in velocity control mode where you are commanding 1.0 Rad/s to joint S0, and you lose network connectivity the robot could result in dangerous motions. By 'timing out', the robot will be safer, reacting to network timeouts, or incorrect control behaviour.

## **A** Joint Position Control

- A** Joint position control mode is the fundamental, basic control mode for Baxter arm motion. In position control mode, we specify joint angles at which we want the joints to achieve. Typically this will consist of seven values, a commanded position for each of the seven joints, resulting in a full description of the arm configuration.
- A** JointCommand.msg mode: `POSITION_MODE`
- A** This joint command is then subscribed to by Baxter's Motor Controller. The motor controller processes this joint command ensuring safety (collision avoidance and detection) and expected behaviour by making the following modifications:



### **Delta Scaling:**

Scale setpoint based on which joint is going to take the longest to achieve. Allows all joints to arrive simultaneously.

### **Velocity Scaling:**

'Speed Ratio' describes the overall velocity scaling.

### **High Speed Scaling:**

High speed scaling reduces execution speed when commanded speed exceeds a high speed velocity threshold **and** the arm's high-speed collision links are in collision.

### **Collision Avoidance:**

Applies offsets to joint commands based on depth of intersection between arm collision geometries and the opposing arm or torso.

### **Joint Limit Clipping:**

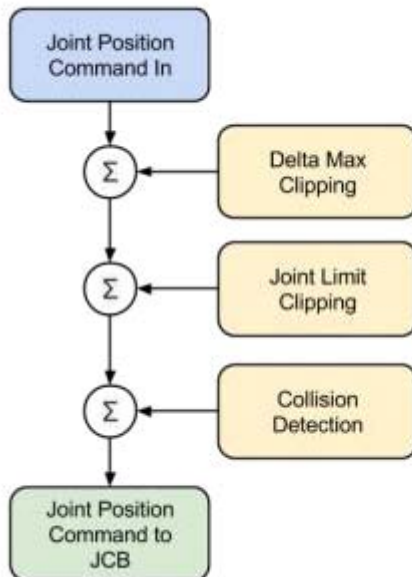
If the joint command is beyond limits, clip the command to respect joint limits.

### **Collision Detection:**

If collision (impact) is detected, set position command to hold current compensating for the impact.

## **A Raw Joint Position Control**

- A** Raw Joint position control mode was introduced to provide a much more direct position control method. The same basic idea as the standard joint position control mode still holds. However, the joint commands are largely left to the JCBs for execution.
- A** JointCommand.msg mode: `RAW_POSITION_MODE`
- A** Advanced Control Mode: As we no longer apply collision avoidance offsets, and allow for very fast motions at the joints velocity limits, this mode should be used with care.
- A** The 'raw' joint position commands only perform the following modification (typically these additions are none):



### **Delta Max Clipping:**

The joint command will be clipped based on the delta max (offset from current position defined by max joint velocity)

### **Joint Limit Clipping:**

If the joint command is beyond limits, clip the command to respect joint limits.

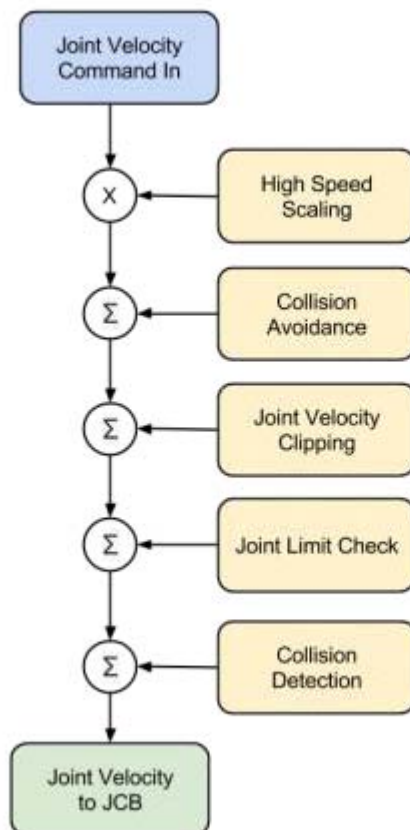
### **Collision Detection:**

If collision (impact) is detected, set position command to hold current compensating for the impact.



## **A** Joint Velocity Control

- A** Joint velocity control mode is an advanced control mode. In velocity control mode, we specify joint velocities at which we want the joints to simultaneously achieve. Typically this will consist of seven values, a commanded velocity for each of the seven joints.
- A** JointCommand.msg mode: `VELOCITY_MODE`
- A** When commanding joint velocities, if a commanded velocity to one of the joints will result in a joint position that is beyond the joints limits, no joints will be commanded, as all of the command is considered invalid. Reason we do this; an example of a common control method, using the Jacobian for Cartesian control resulting in joint velocity commands. If a single joint hits its limit, the rest of the joints will still be commanded, resulting in obscure and potentially dangerous motions. We recommend either implementing a joint space potential field or joint limit check before submitting the joint velocity commands.
- A** Advanced Control Mode. As we allow for very fast joint velocity up to the joint velocity limits, this mode should be used with care.
- A** This joint command is subscribed to by Baxter's Motor Controller. The motor controller processes this joint velocity command (applying collision avoidance and detection) and expected behaviour by making the following modifications:



### **High Speed Scaling:**

High speed scaling reduces execution speed when commanded speed exceeds a high speed velocity threshold **and** the arm's high-speed collision links are in collision.

### **Collision Avoidance:**

Applies offsets to joint commands based on depth of intersection between arm collision geometries and the opposing arm or torso.

### **Joint Velocity Clipping:**

Limits joint velocity command to not exceed maximum joint velocities.

### **Joint Limit Check:**

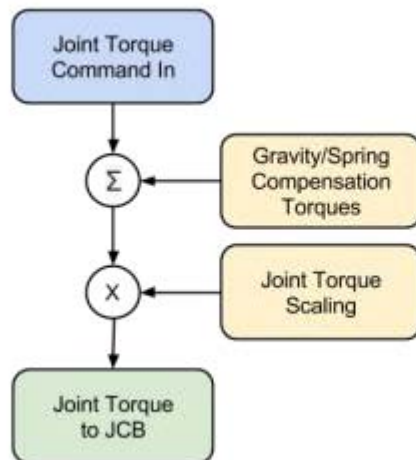
Validates that resulting joint position will be within joint limits. If not, no velocity will be commanded to any joint.

### **Collision Detection:**

If collision (impact) is detected, set position command to hold current compensating for the impact.

## **A** Joint Torque Control

- A** Joint torque control mode is an advanced control mode. In torque control mode, we specify joint torques at which we want the joints to simultaneously achieve. Typically this will consist of seven values, a commanded torque for each of the seven joints.
- A** JointCommand.msg mode: TORQUE\_MODE
- A** Joint torque commands are applied in addition to gravity and spring compensation torques. This default can be disabled by publishing an [std\\_msgs/Empty](#) message on the topic:  
`/robot/limb/right/suppress_gravity_compensation` at a rate > 5Hz.
- A** Advanced Control Mode. When commanding in torque control mode, access is granted to the lowest control levels. This puts much responsibility on the control program and should be used with care.
- A** This joint command is then subscribed to by Baxter's Motor Controller. The motor controller processes this joint torque command (applying collision avoidance and detection) and expected behavior by making the following modifications:



### **Gravity/Spring Compensation:**

The joint torque command is applied in addition to the gravity and S1 spring compensation torques.

### **Joint Torque Scaling:**

Scales all joint torques if a torque command exceeds the maximum allowable torque for that joint. This scaling ratio is defined as  $\text{torque\_max} / \text{torque\_command}$ .

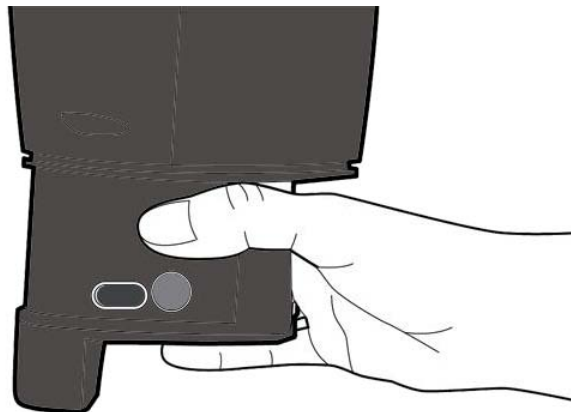


## **A Gravity Compensation Torques**












- A** In order to oppose the effect of gravitational force acting across Baxter's arm, gravity compensation torques have to be applied across that arm. This is a basic mode that is active from the time the robot is enabled. The built-in gravity compensating model uses [KDL](#) for calculating the gravity compensation torques. The springs attached to S1 also require compensation torques, these are found using an internal spring model and applied in conjunction with the torques from KDL. These gravity compensation torques are passed directly to the JCB through a separate channel and are applied irrespective of the controller being active.
- A** The gravity compensation torques are available via the topic `robot/limb/<side>/gravity_compensation_torques` of message type `baxter_core_msgs/SEAJointState`
- A** In order to disable the gravity compensation torques, an empty message should be published to the topic `/robot/limb/<side>/suppress_gravity_compensation` at greater than 5 Hz.

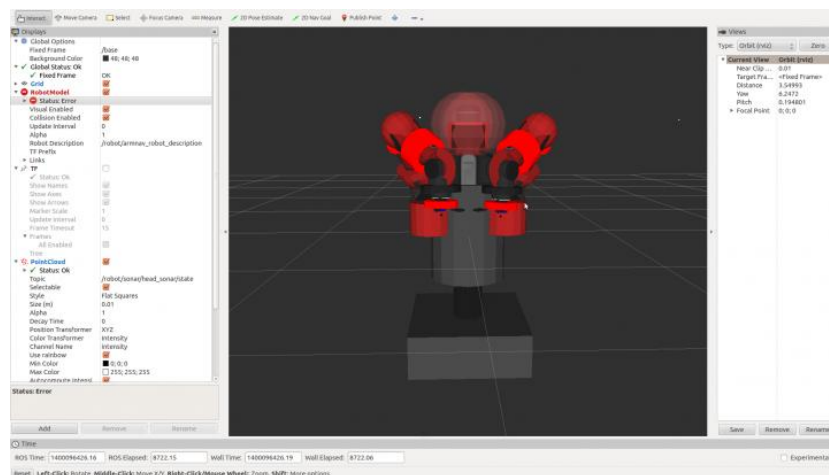
## **A Zero-G**

- A** 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 the workspace. 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.
- A** The Zero-G mode can be enabled by grasping the cuff over its groove as below



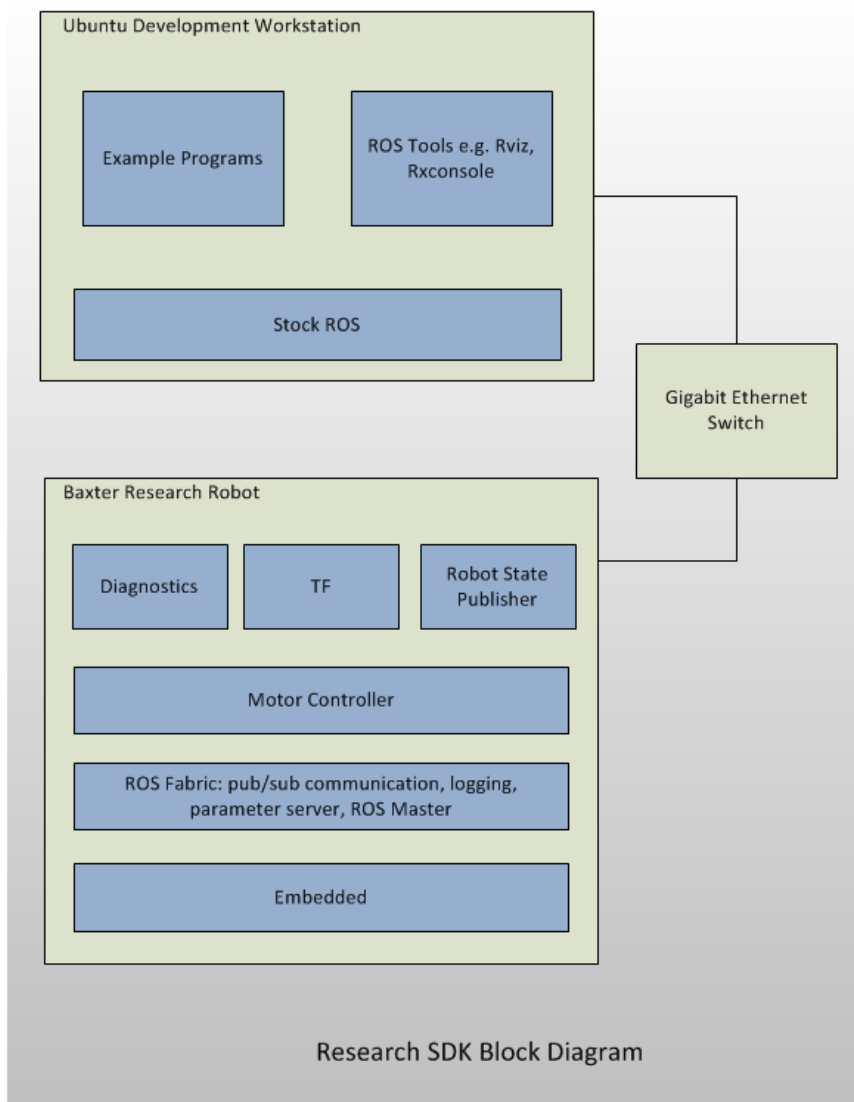
## Collision Model

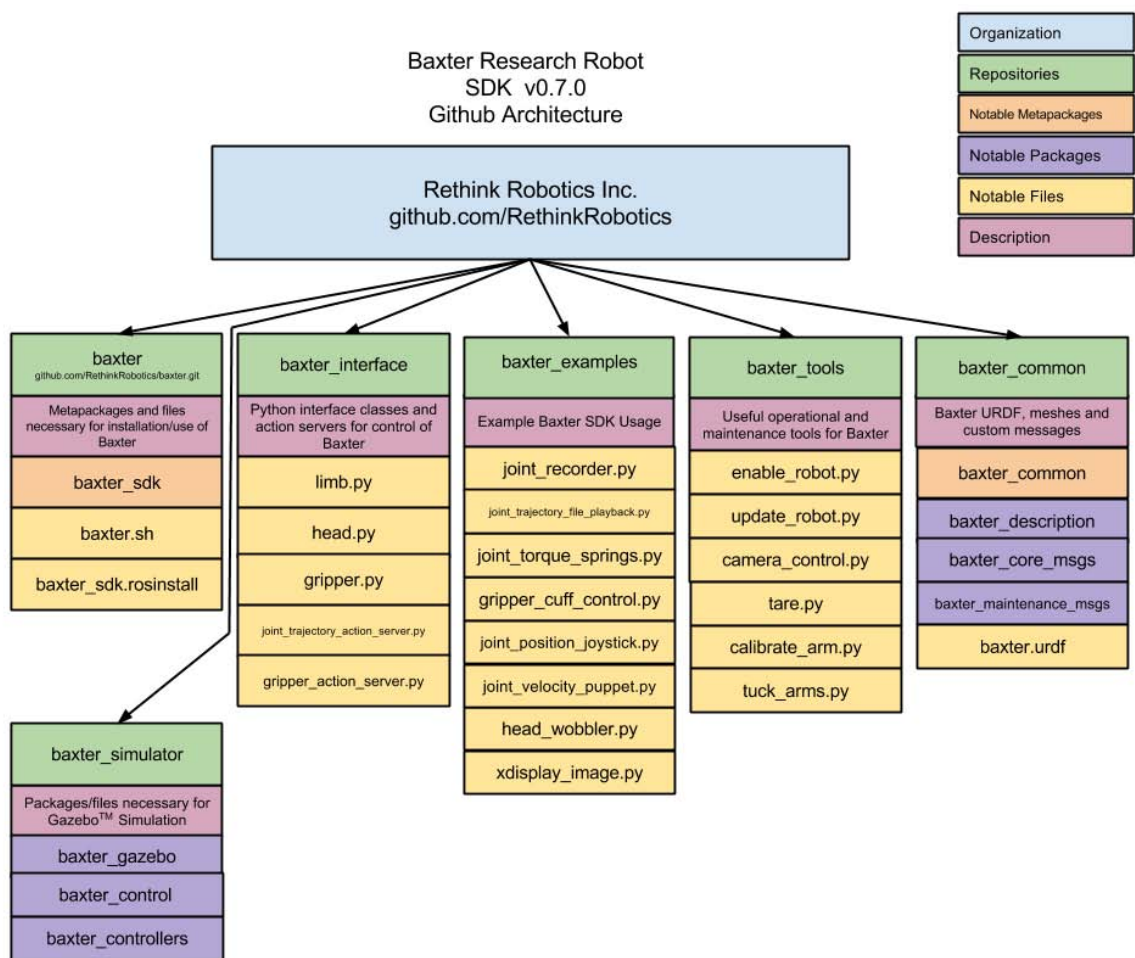
-  There are a total of three levels of collision models in Baxter.
  
-  Each of Baxter's links has its own collision block which is slightly bigger than the size of that link. When these block comes into contact with each other, collision avoidance model is triggered.
  -  The collision blocks can be visualized in RVIZ by setting the Robot Description field under RobotModel as `"/robot/armnav_robot_description"` and enabling the field Collision Enabled.
-  The next collision model involves the detection of two types of collisions - impact and squish.
  -  Impact is detected when a sudden change in torque is sensed across any of the joint. This can be related to a scenario in which a moving hand collides with a human. Here, the sudden change in torque is sensed during the impact and the robot comes to a stop for 2 seconds before attempting to move again.
  -  Squish is detected when joint tries to press against a stationary object. For instance, when a robot arm tries to push a wall, the torque applied across the joint increases and it immediately stops when the applied torque is greater than a threshold. It resumes its motion after 2 seconds.
-  The final collision model is high speed scaling. This is to avoid the collisions between two limbs moving at higher speeds.
  -  The collision blocks for this purpose are larger than the original collision blocks.
  -  The collision detection happens when the speed of the limb in Cartesian space is higher than 0.2 m/s.
  -  In order to avoid the collision, the corresponding velocities are scaled down and converted back as position commands.
-  To disable the collision avoidance on a given arm, you need to publish an empty message at greater than 5hz to the topic, `/robot/limb/<side>/suppress_collision_avoidance`



# Software Architecture Overview:

- A The Baxter Research Robot SDK provides a software interface allowing researchers of all disciplines to develop custom applications to run on the Baxter platform.
- A 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](#).
- A The following diagrams show the software architecture:

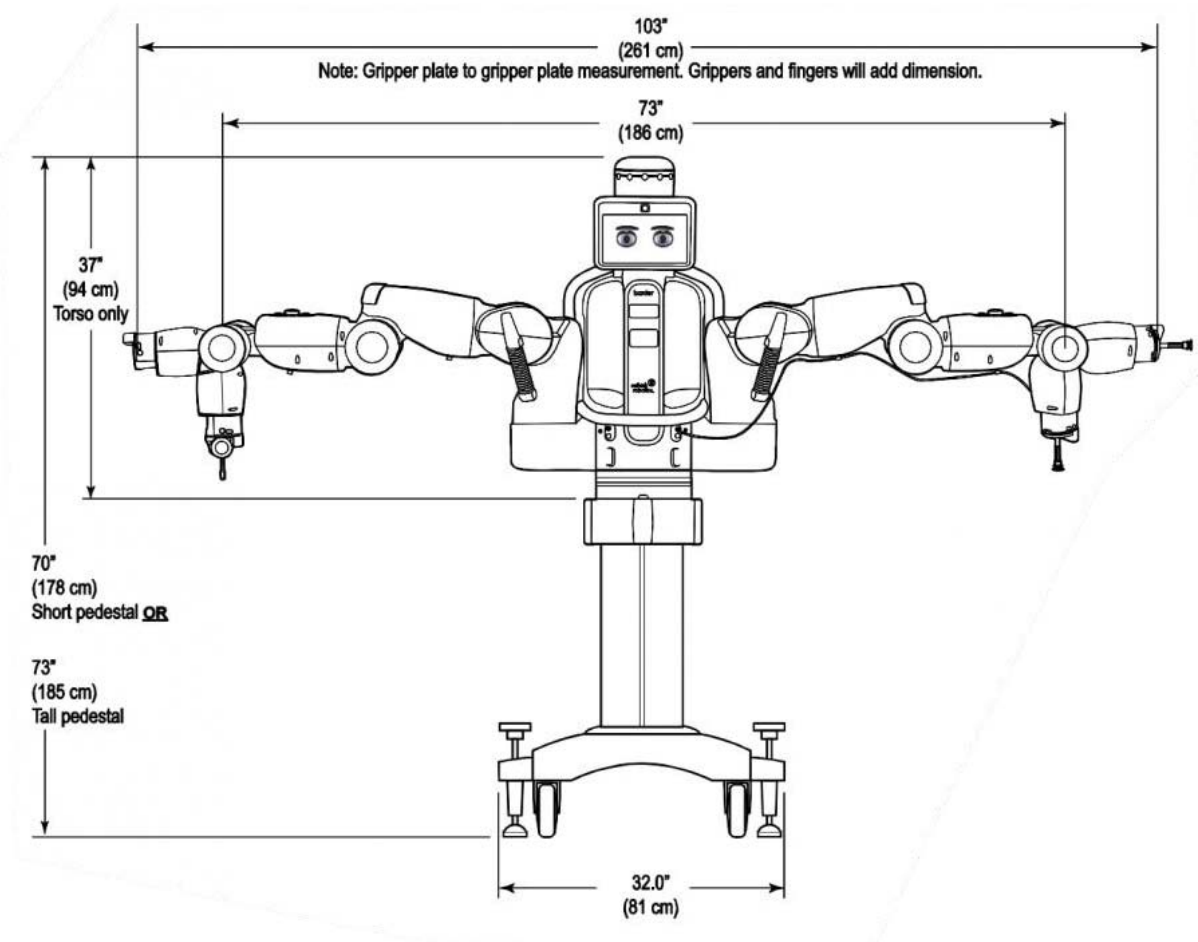




# Hardware Specification:

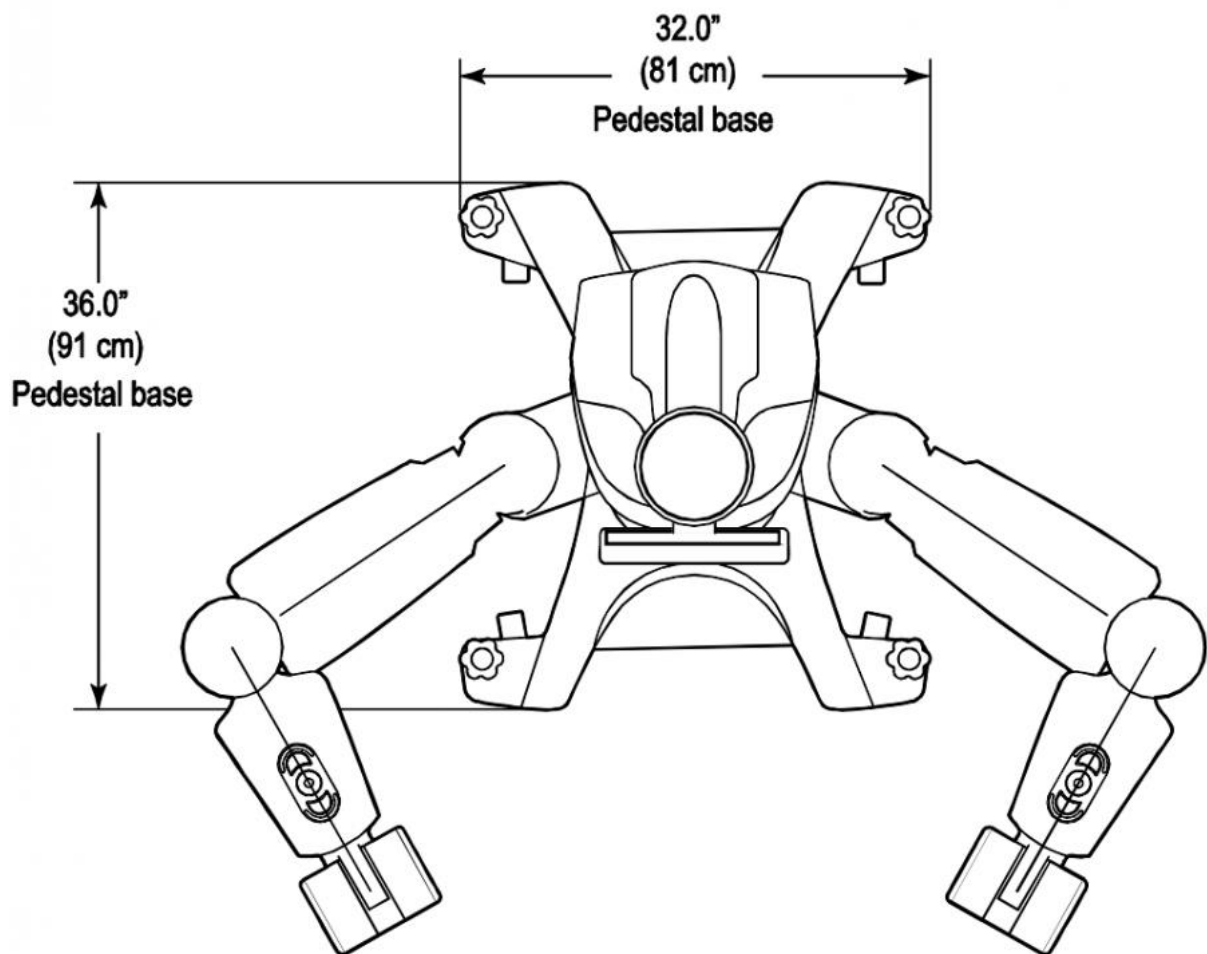
## A General Dimensions

### A Front



## General Dimensions

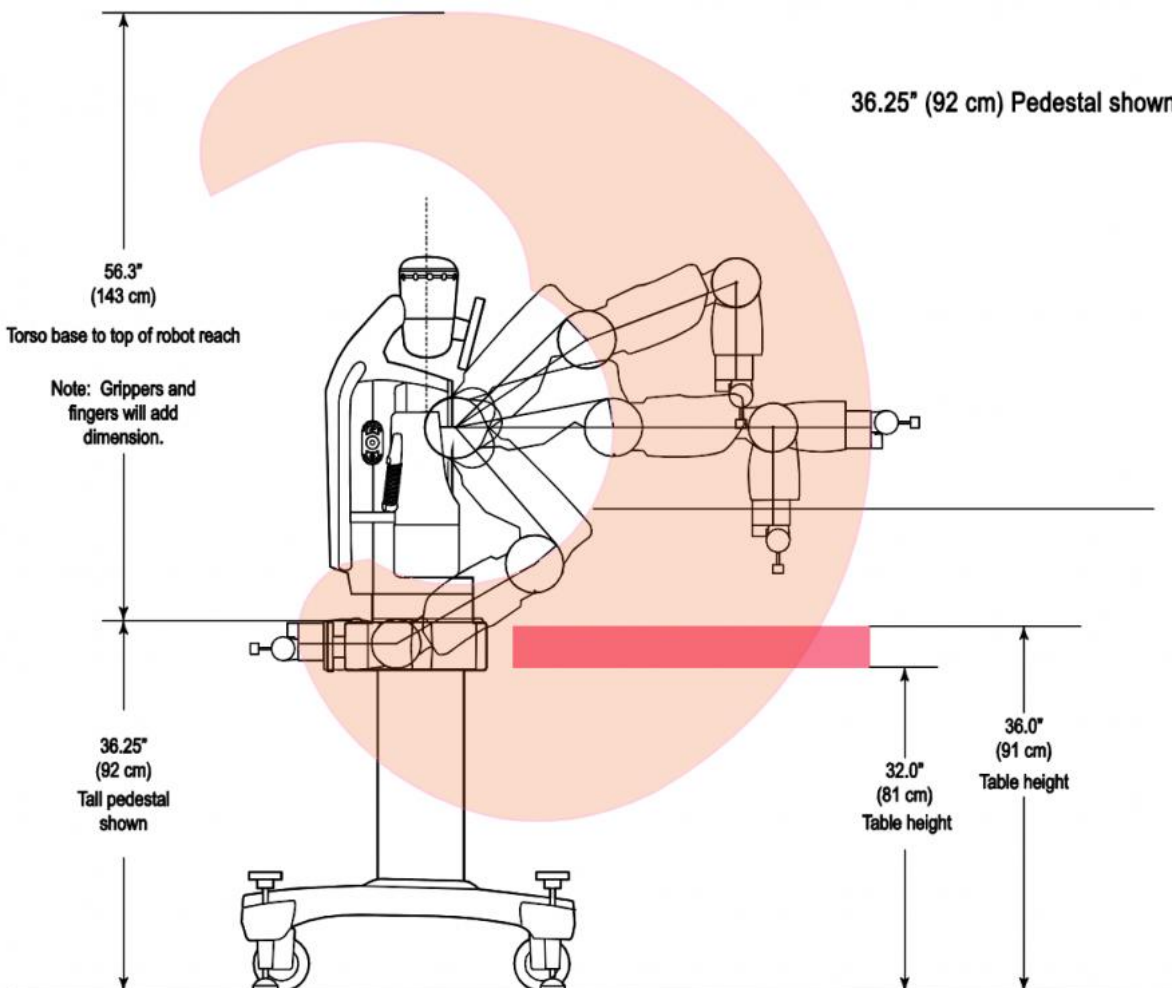
### Top





## Workspace Reach Dimensions

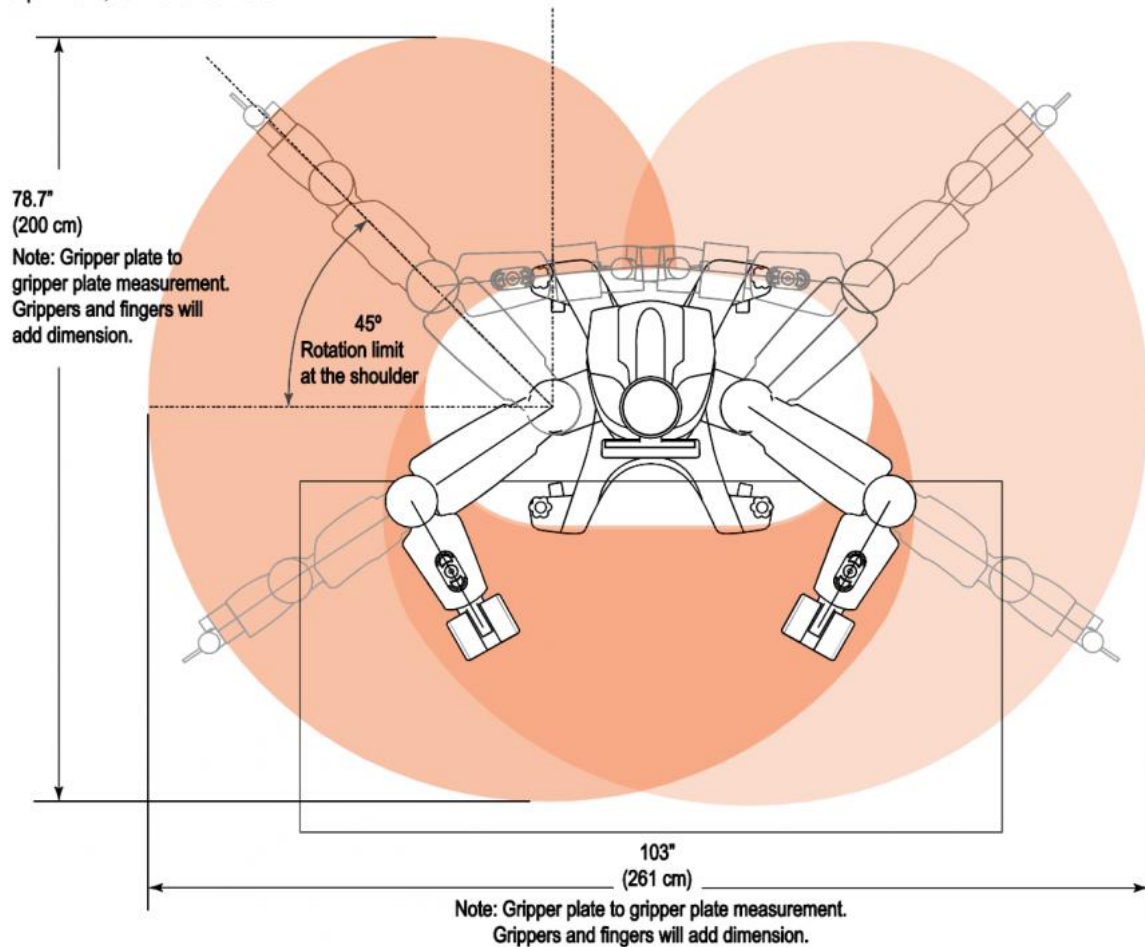
### Side



## Workspace Reach Dimensions

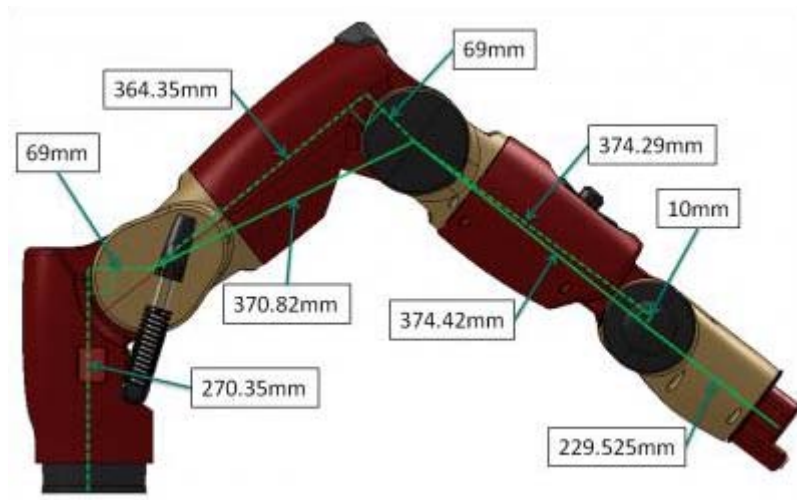
### Top

Top view, arms extended

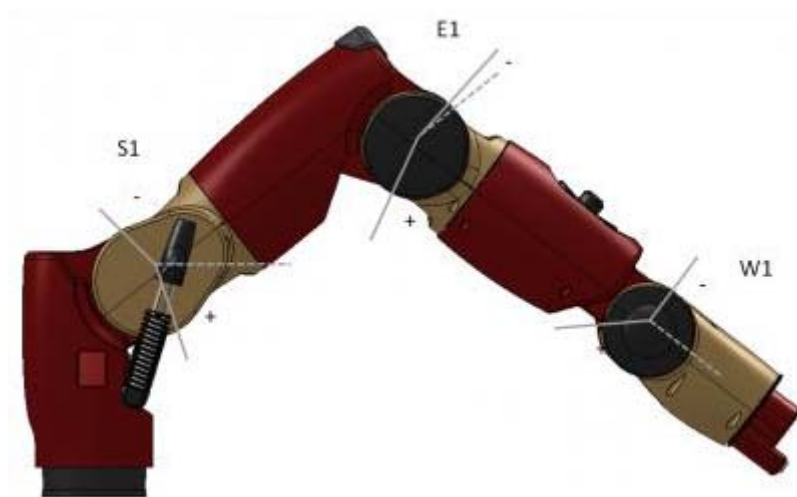


## Arm Physical Constraints

### Arm Kinematic Link Lengths

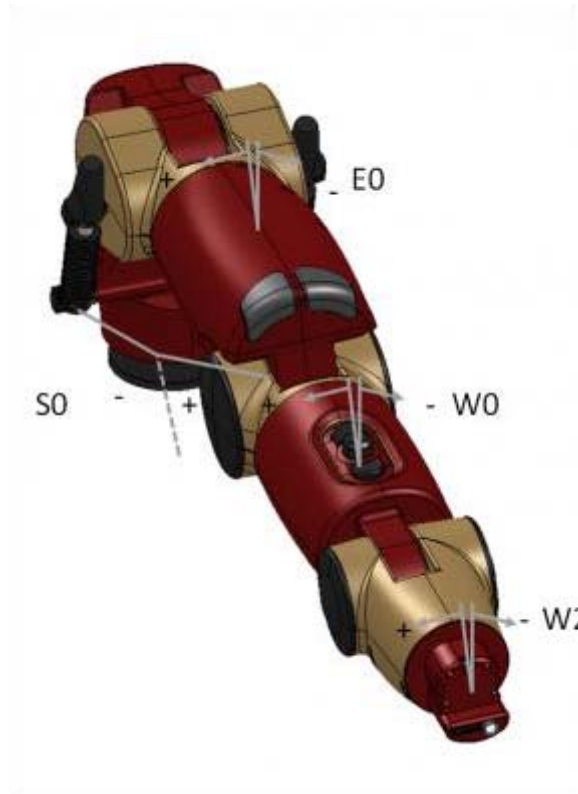


### Arm Kinematic Range of Motion – Bend



Joint	Range (Degrees)	Range (Radians)
S1	+60,-123:183	+1.047,-2.147:3.194
E1	+150,-3:153	+2.618,-0.052:2.67
W1	+120,-90:210	+2.094,-1.571:3.665

## **A** Arm Kinematic Range of Motion – Twist



Joint	Range (Degrees)	Range (Radians)
E0	+173.5,-173.5:347	+3.028,-3.028:6.056
S0	+51,-141:192	+0.890,-2.461:3.351
W0	+175.25,-175.25:350.5	+3.059,-3.059:6.117
W2	+175.25,-175.25:350.5	+3.059,-3.059:6.117

## **A** Joint Feedback Sensor Resolution

- A** The resolution for the joint sensors is 14 bits (over 360 degrees); giving  $360/(2^{14}) = 0.021972656$  degrees per tick resolution.
- A** All joints have a sinusoidal non-linearity, giving a typical accuracy on the order of  $\pm 0.10$  degrees, worst case  $\pm 0.25$  degrees accuracy when approaching joint limits. In addition, there may be an absolute zero-offset of up to  $\pm 0.10$  degree when the arm is not calibrated.

#### Joint Peak Torque

Joint	Peak Torque
S0,S1,E0,E1	50Nm
W0,W1,W2	15Nm

#### Positional Accuracy

Type	Accuracy
Whole Workspace	+/-5mm
Limited Envelope	+/-0.5mm

#### Payload

Type	Accuracy
Max Payload (inc End-Effector)(Safety Enabled)	2.3kg
Max Payload (inc End-Effector)(Safety Disabled)	~25kg

#### Onboard Computation

Description	Specification
Processor	3rd Gen Intel Core i7-3770 Processor (8MB, 3.4GHz) w/HD4000 Graphics
Memory	4GB, NON-ECC, 1600MHZ DDR3
Hard Drive	128GB Solid State Drive

#### Camera and Screen Specifications

Description	Specification
Max Resolution	1280 x 800 pixels
Effective Resolution	640 x 400 pixels
Frame Rate	30 frames per second
Focal Length	1.2mm
Screen Resolution	1024 x 600 pixels

## Power

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