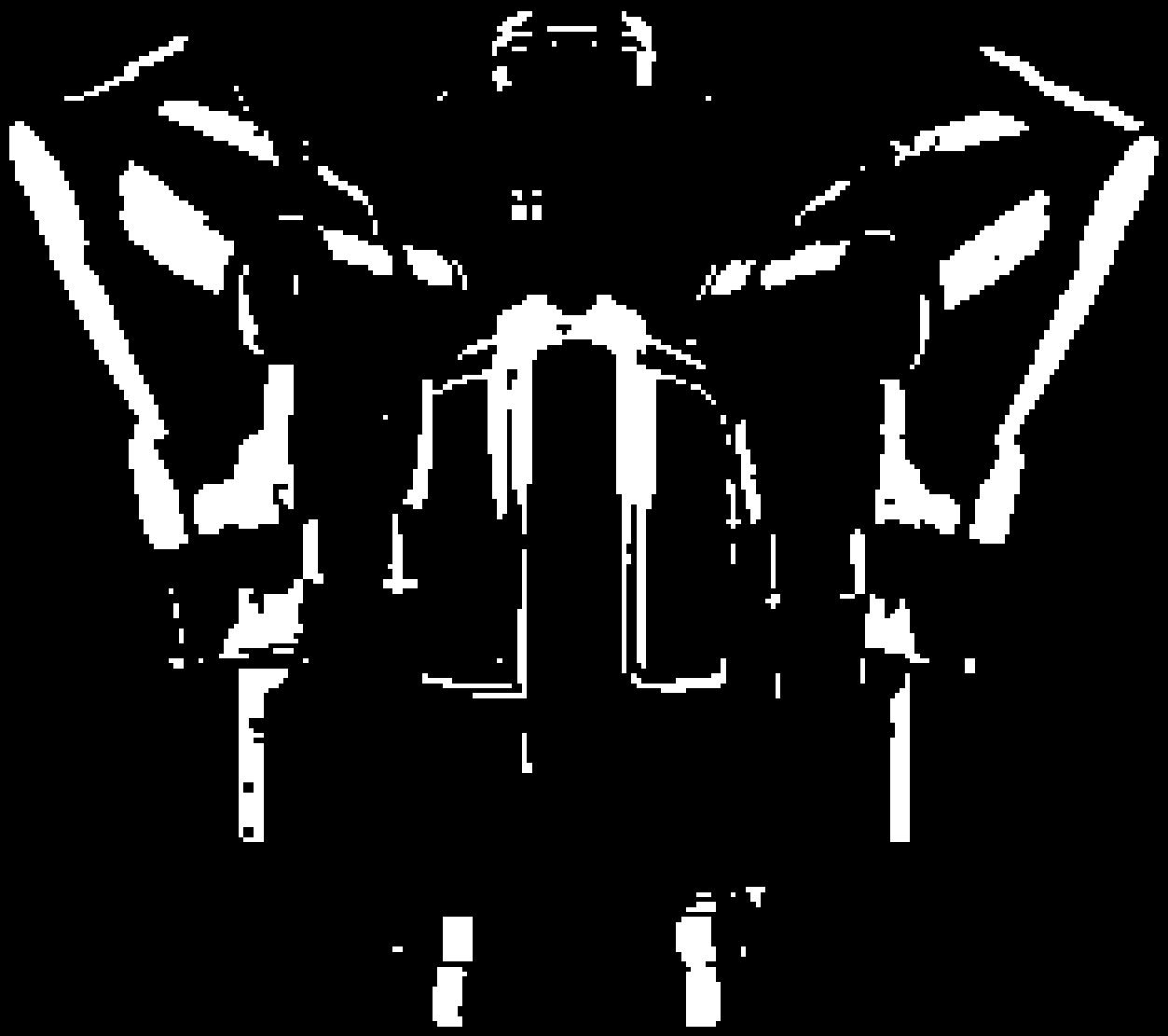


Technical Specification Datasheet &
Hardware Architecture Overview

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: https://api.rethinkrobotics.com/

Hardware Architecture Overview:

Arms:

Baxter has two seven degree-of-freedom (DOF) arms. Seven DOF arms are chosen because 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 the new 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/drl/wiki/images/3/3f/Williamson_mit_thesis.pdf

Fig.

Joint Naming Convention Baxter's joints are named using the following convention.

From the base, along the arm to the End-Effector:

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

Joint Control

Fundamental to the operation of Baxter via the SDK is a hierarchy with the available joint control modes. The joint control modes define the 'mode' in which we can control Baxter's limbs in joint space. These are provided via a ROS message API.

When an joint position command is published from the development PC, the 'xdisplay' process which represents a motor control plugin subscribes to this message. This message is then parsed, and stored in memory based on control mode.

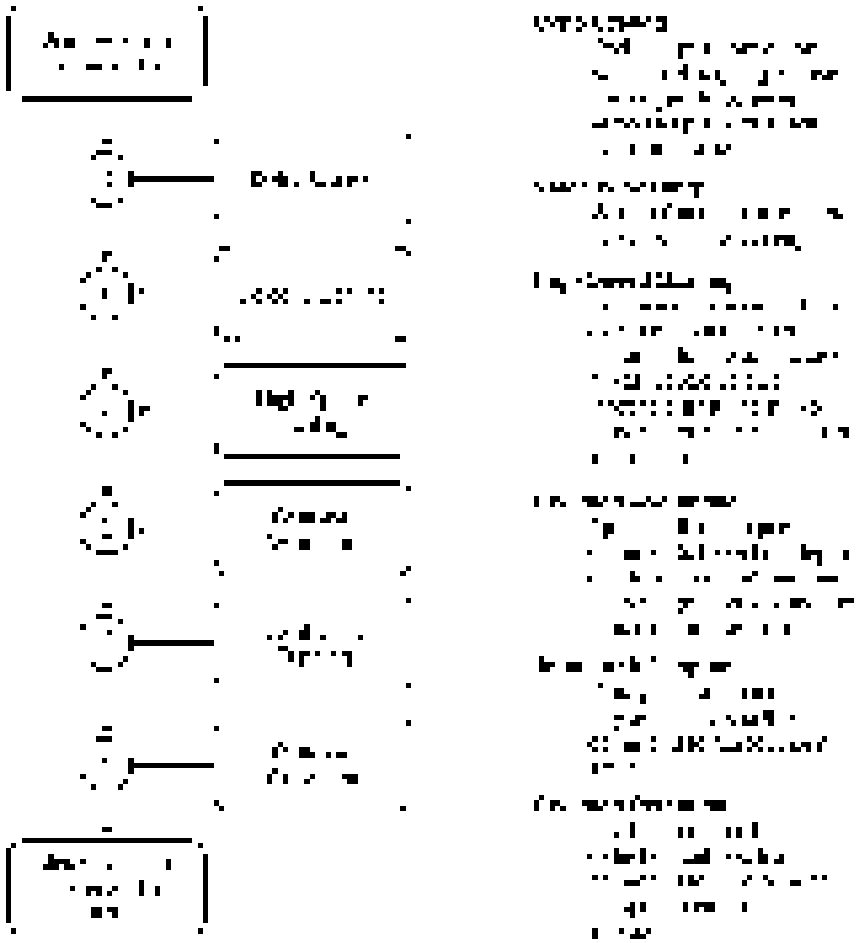
Depending on the control mode, modifications are made to the input commands. These modifications are typically due to the safety control layer (e.g. arm-to-arm collision avoidance, collision detection, etc.)

A control rate timeout is also enforced at this motor controller layer. This states that if a new 'JointCmd' 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 set 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 3.0 rad/s to joint S0, and you lose network communication by the robot could result in dangerous motions. By 'timing out', the robot will stop after, reacting to network timeouts, or incorrect control behaviour.

Joint Position Control

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. JointCommand msg mode: POSITION_MODE.

This joint command is then published to Baxter's Motor Controller. The motor controller processes the joint command and ramps it by location and checks and detects if and expected behaviour by making the following modifications:



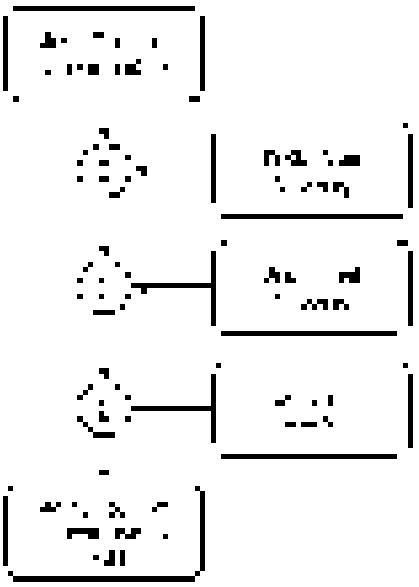
New Joint Position Control

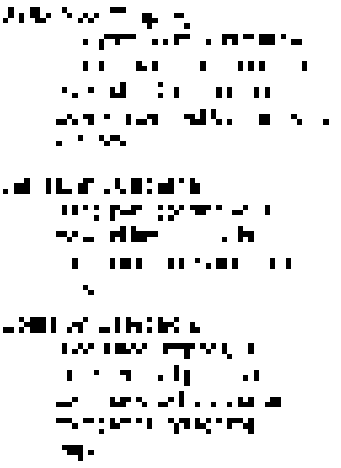
New Joint position control mode was introduced to provide a much more direct position control method. This is the best choice for the standard joint position control mode still holds. However, the joint commands are largely left to the ESC for execution.

JointCommand.msg mode: xxx, xxx, xxx

Advanced Control Mode As we no longer apply collision avoidance offsets, and allow for very fast motions at the joints velocity limit, this mode should be used with care.

The new joint position commands only perform the following most limit on (typically these additions are made:





Joint Velocity Control

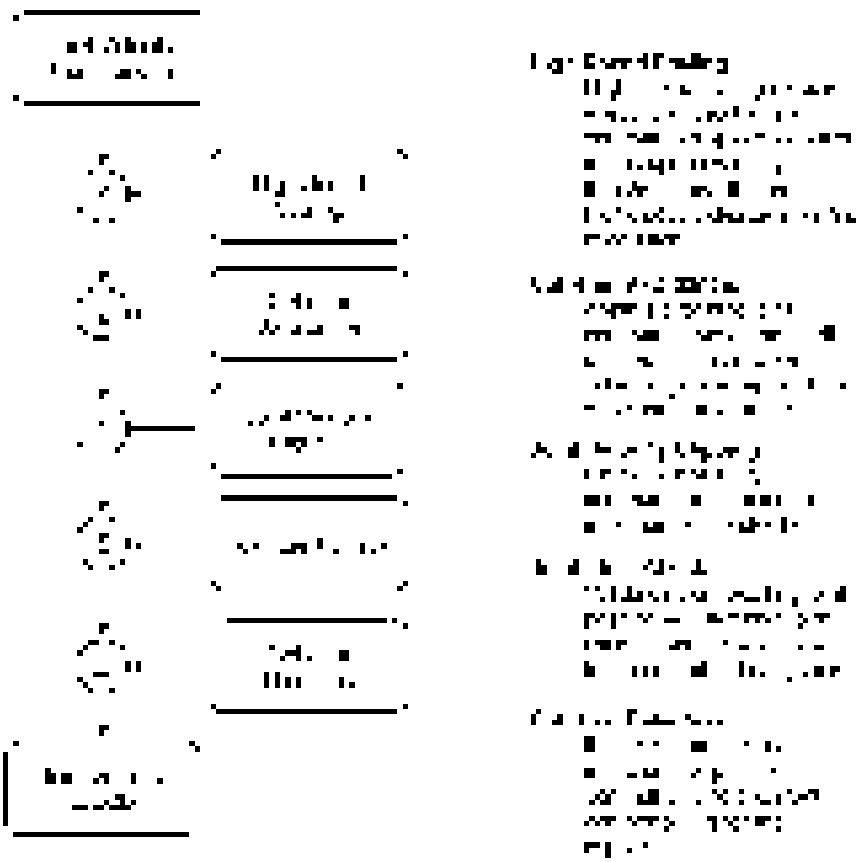
Joint velocity control mode is an advanced control mode. In velocity control mode, we specify joint velocities at which we want the joints to simultaneously achieve. Typically this will be consist of seven values, a commanded velocity for each of the seven joints.

JointCommand msg mode: VELOCITY_MODE

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. Below on mode 1 is an example of a common control method, using the Jacobian for Cartesian control resulting in joint velocity commands. If a single joint hits a 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 checking before calculating the joint velocity commands.

Advanced Control Mode As mentioned for very fast joint velocity up to the joint velocity limits, this mode should be used with care.

This joint command is subscribed to by Baxter's Motor Controller. The motor controller processes this joint velocity command (applying control loops on velocities and others) and executes behaviour by making the following modifications:



Joint Torque Control

Joint torque control mode is an advanced control mode. In torque control mode, we specify joint torques at which we want the joints to already be achieve. Typically this will be consist of seven values, a commanded torque for each of the seven joints.

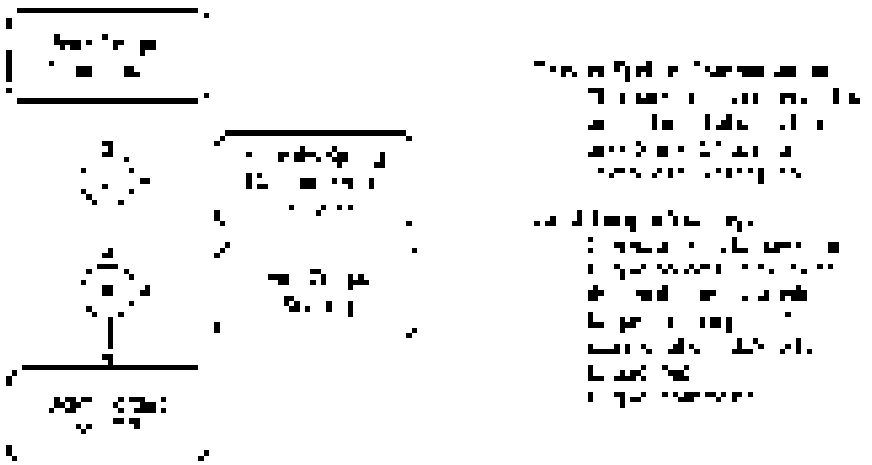
JointCommand.msg mode: COMMAND_MODE.

Joint torque commands are applied in addition to gravity and spring compensation torques. This default can be disabled by publishing on dd_no_gravity message on the topic=

/robot/limb/right/suppress_gravity_compensation and the message data.

Advanced Control Mode. When commanding in torque control mode, access is granted to the lowest control levels. This puts much responsibility on the control programmer and should be used with care.

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 model call once:



Gravity Compensation Torques

In order to oppose the effect of gravitational force acting on the robot's arms, gravity compensation torques have to be applied across the arms. This is a basic mode that is active from the time the robot is enabled. The built-in gravity compensation model uses KDL for calculating the gravity compensation torques. The springs attached to SEA 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 published directly to the JCB through a separate channel and are applied irrespective of the controller being active.

The gravity compensation torques are available via the topic
/robot/limb/right/gravity_compensation_torques of message type baxter_core_msgs/SEAJointState.

In order to disable the gravity compensation torques, an empty message should be published to the topic
/robot/limb/right/suppress_gravity_compensation at greater than 5 Hz.

Zero-G

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 control laws are disabled and so the arms can be freely moved across the work space. 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 control laws (as they would not be active, and so the arms can be moved freely around, hence the name). The Zero-G mode can be enabled by grasping the cuff over the grooves 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 that link. When these blocks come 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/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 arm collides with a human. Here, this sudden change in torque is sensed during the impact and the robot comes to a stop for 3 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 remains fixed in place. When the applied torque is greater than a threshold it resumes its motion after 3 seconds.

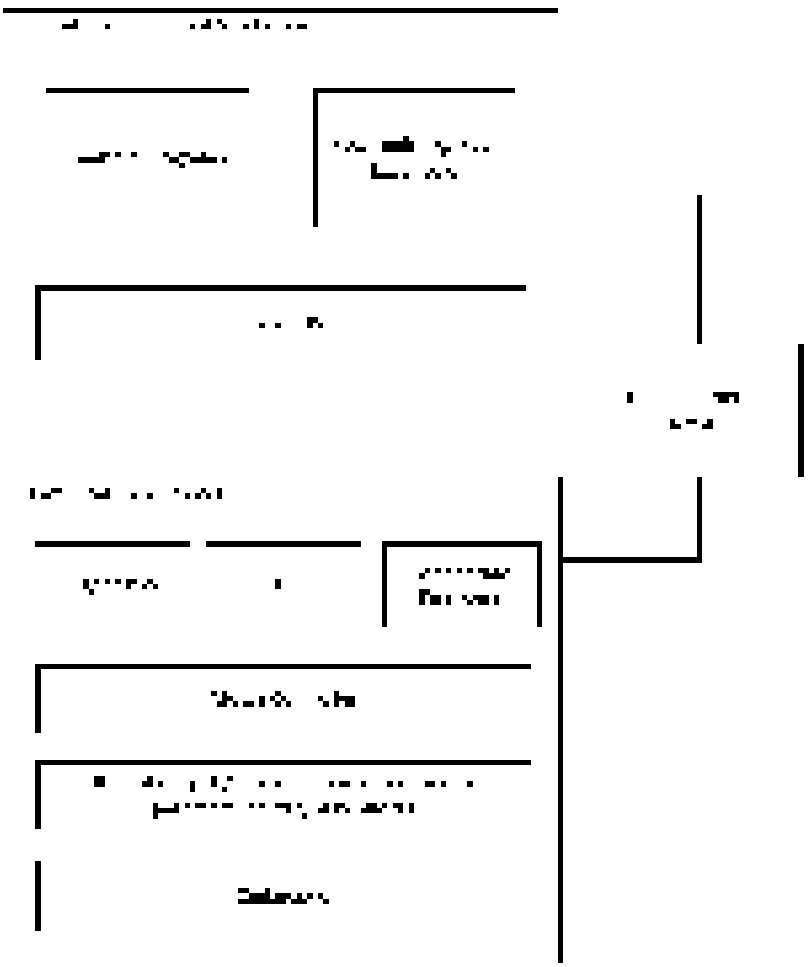
The final collision model is high speed sensing. This is to avoid the collision between two links moving at higher speeds.

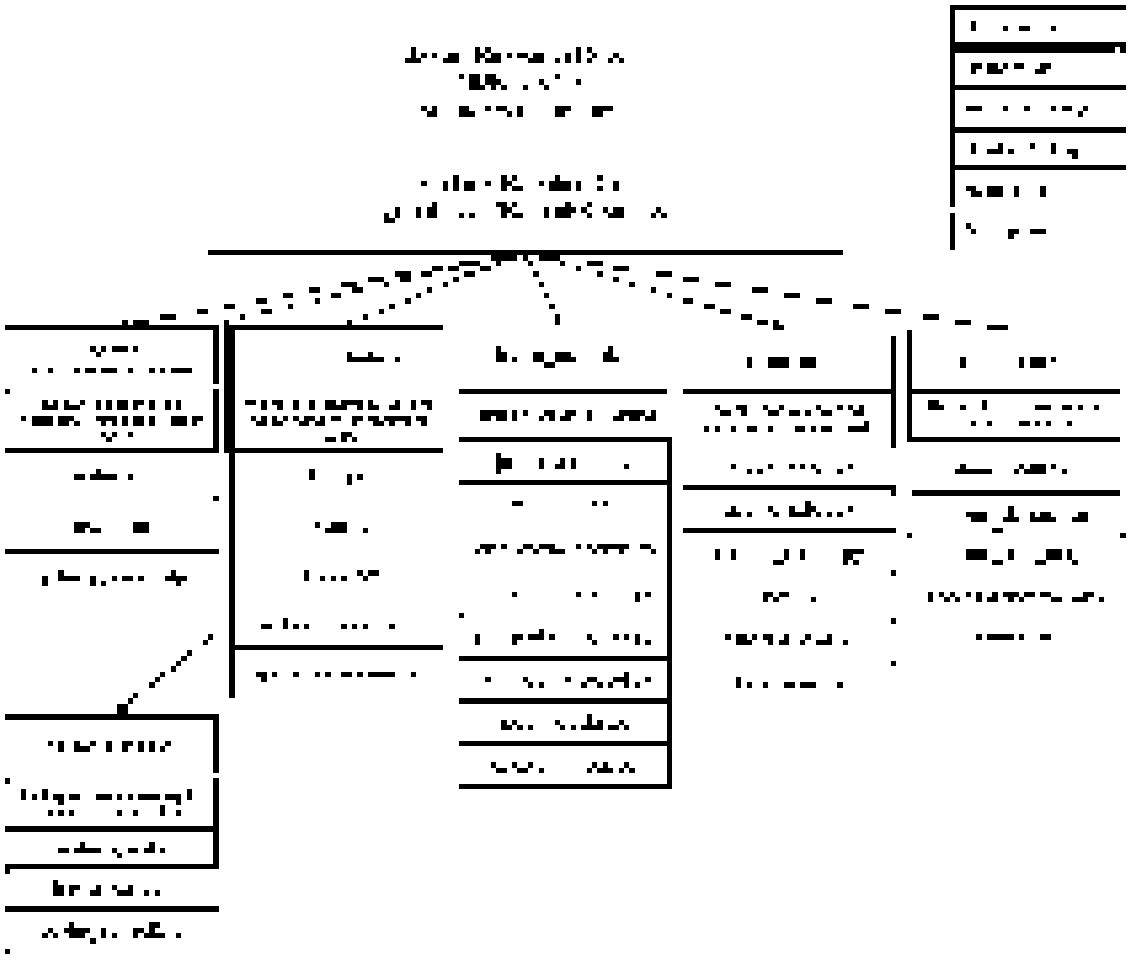
- The collision blocks for this purpose are larger than the original collision blocks.
- The collision detection happens when the speed of the links in Cartesian space is higher than 0.3 m/s.
- In order to avoid the collision, the corresponding velocities are scaled down and converted back as position commands.

To disable the collision avoidance on a given arm, you need to publish an empty message at greater than 2hz to the topic:

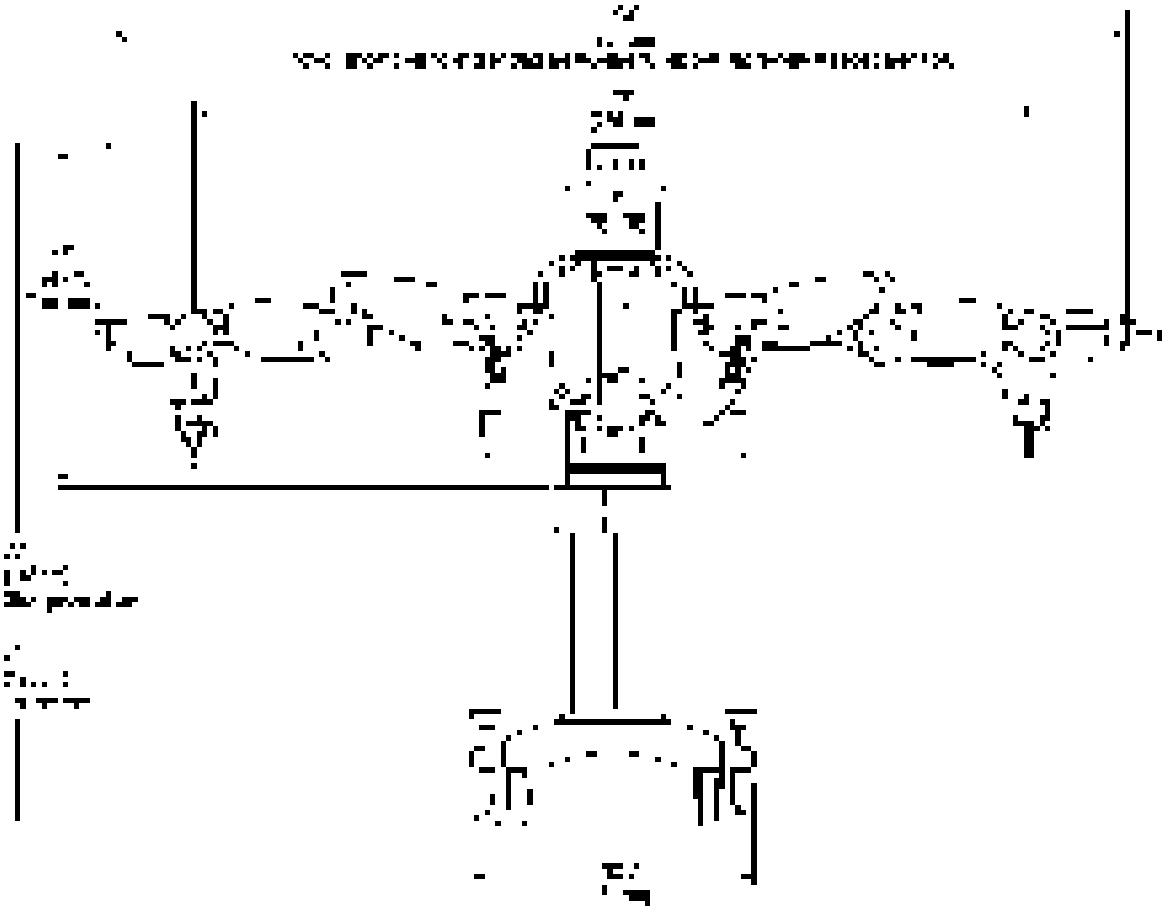
Software Architecture Overview:

The Hexter Research Robot SDK provides a software interface allowing research and
educational users to develop custom applications to run on the Hexter platform.
The SDK interfaces with the Hexter Research Robot via RS-485 (Robot Operating
System) Hexter provides a stand-alone RDS Master to which any development
workstation can connect and control Hexter via the various RS-485 APIs.
The following diagram shows the software architecture:



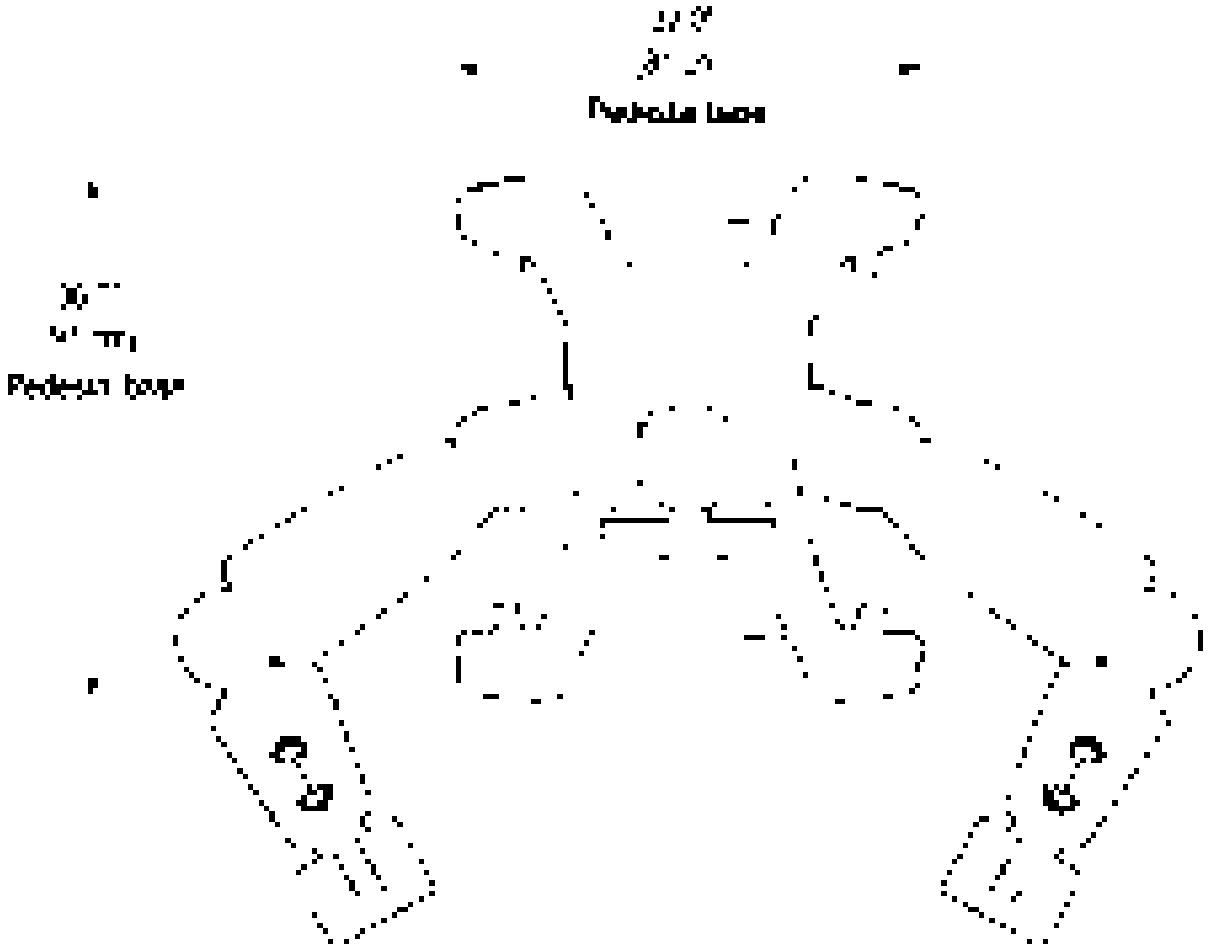


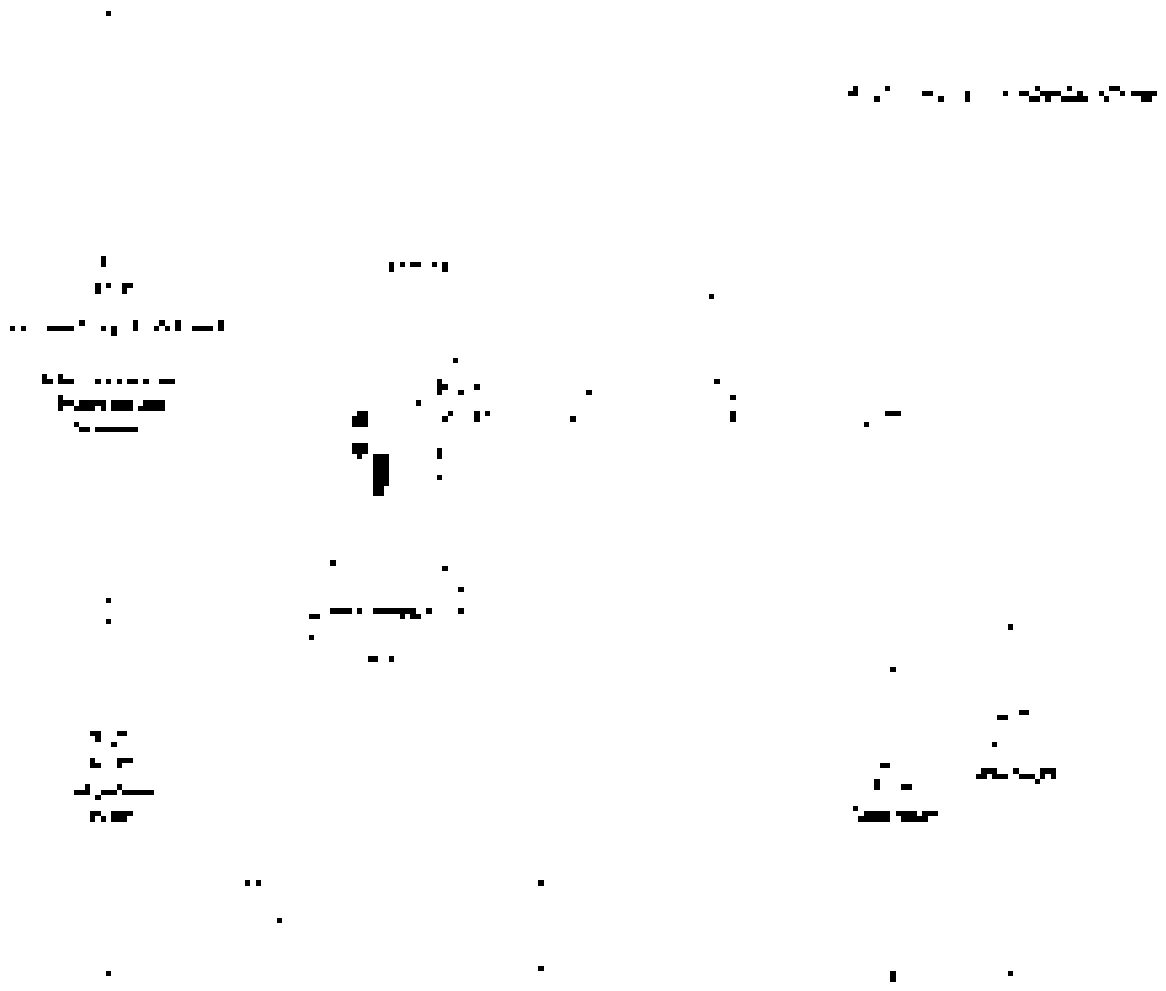
Hardware Specification :

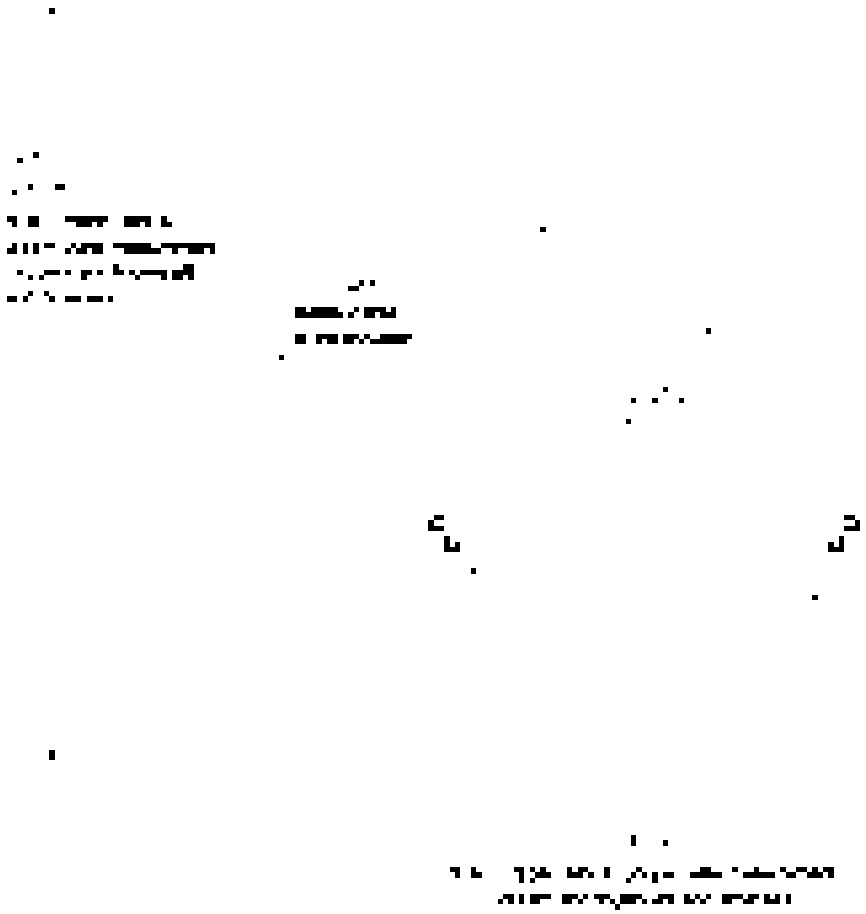


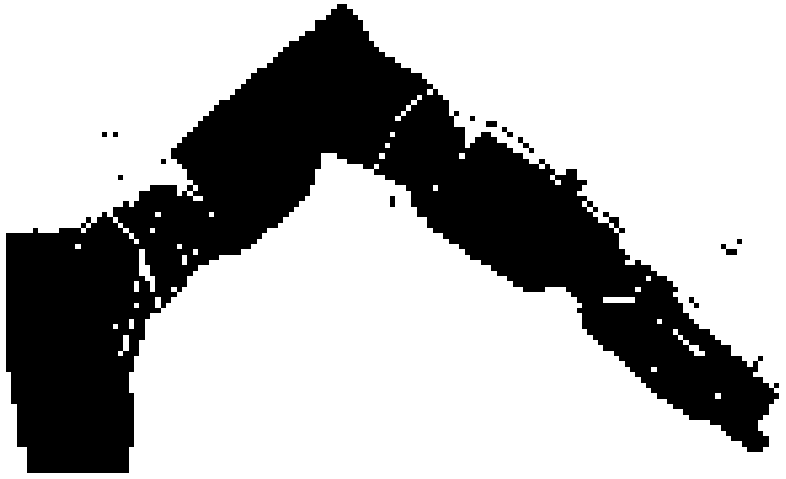
General Dimensions

Top









[illegible]	[illegible]	[illegible]
J1	[illegible]	[illegible]
J2	[illegible]	[illegible]
J3	[illegible]	[illegible]

Arm Kinematic Range of Motion – Twist

Joint	Range (Degrees)	Range (Radians)
E0	+174.987, -174.987, 349.97	+3.0541, -3.0541, 6.108
S0	[illegible]	[illegible]
W0	+175.25, -175.25, 350.5	+3.059, -3.059, 6.117
W2	[illegible]	[illegible]

Joint Position Sensor Resolution

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

Rated Peak Torque

Joints	Peak Torque
[illegible]	[illegible]
W0, W1, W2	[illegible]

Positional Accuracy

Type	Accuracy
[illegible]	[illegible]
Unload Envelope	[illegible]

Payload

Type	Accuracy
Max Payload (Rigid End Effector) [illegible]	[illegible]
Max Payload (Incl End Effector's Safety Enabled)	[illegible]

Backend Computation

Description	Specification
Processor	[illegible]
Memory	[illegible]
Hard Drive	[illegible]

Sensors and Screen Specifications

Description	Specification
Max Resolution	[illegible]
Effective Resolution	[illegible]
Frame Rate	[illegible]
Focal Length	[illegible]
Screen Resolution	[illegible]

Power

Description	Specification
Auxiliary Operation	DC-to-120W AC Inverter (Note: the Baxter robot has an internal PC, which cannot be powered directly off of 24V DC)
Interface	[illegible]
Wall Power Consumption Electrical Efficiency	Max: 520VAC, 720W max per unit [illegible]
Power Supply	Uses rapid hot-swap DC switching power supply for robot power bus
Tolerance to Surge	[illegible]
Hot Swap Window Voltage Ambulance	Hot swap time: 30ms [illegible]