ROS – Baxter, Simulation and Hands-on

Helge G. Grøn helgegg@mp.aau.dk

AALBORG UNIVERSITY

Say hello to BAXTER!

What is a collaborative robot (cobot)?

A cobot or co-robot (from collaborative robot) is a robot intended to physically interact with humans in a shared workspace

Examples of cobots :

- BAXTER (Rethink Robotics)
- UR series (Universal Robots)
- LBR iiwa (KUKA)
- Justin (DLR)
- COMAN Robot (IIT)
- YuMi (ABB)
- Etc.

The BAXTER robot

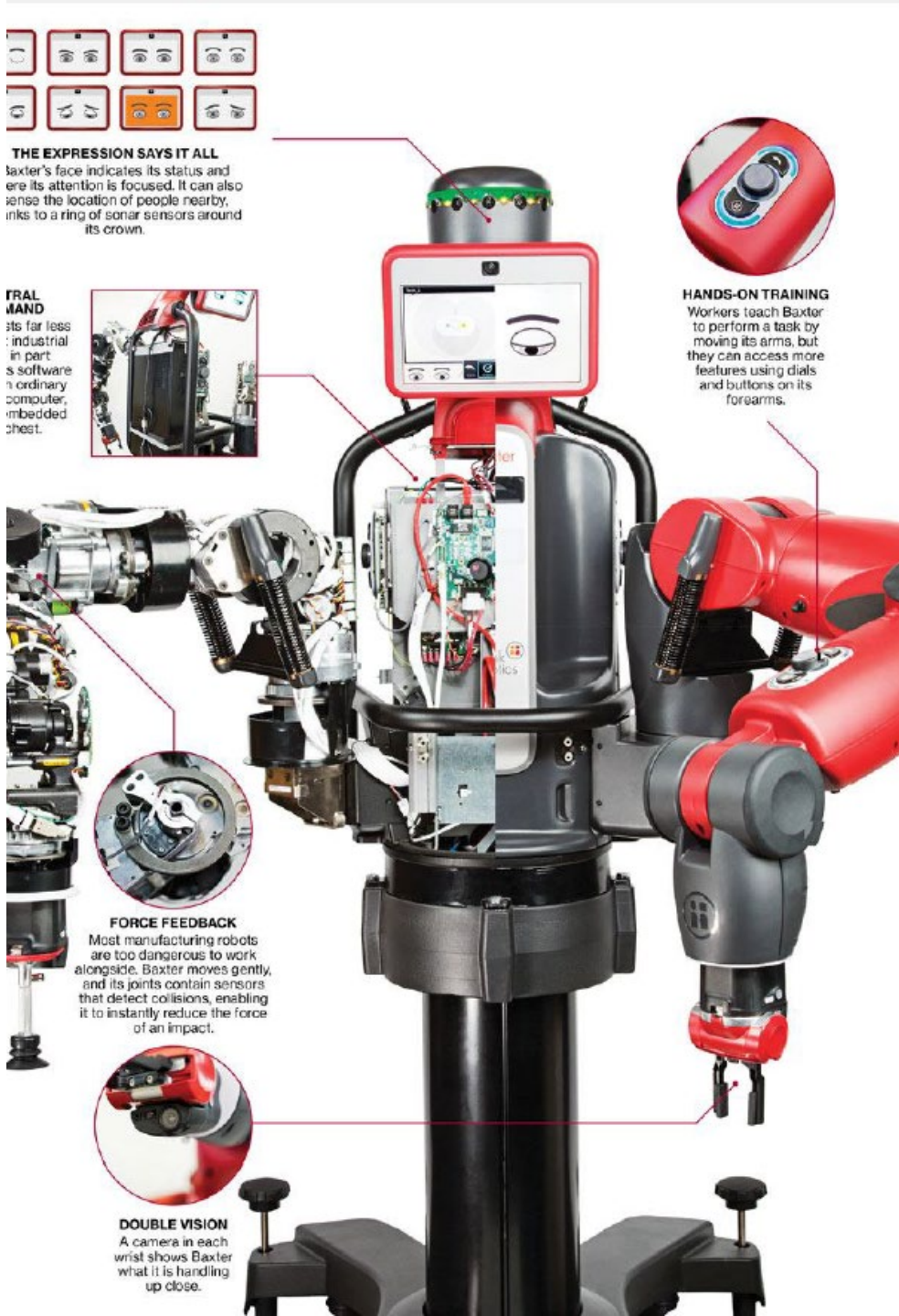
Hello!

I am BAXTER.

I am a safe, flexible collaborative robot.

I can manipulate items up to **2.2Kg** (including the EOAT) with an accuracy of **+/- 5mm**.

I have two arms (**7 DOF each**) that can work independently, and I can also rotate my head and nod.



BAXTER's modes of interaction

- 3 cameras (one on its head and one on each cuff - 640 x 400 px, 30fps)
- IR range sensors (one on each cuff, range from 4 to 40cm)
- Accelerometer (one on each cuff)
- Touch sensors (one pair on each cuff)

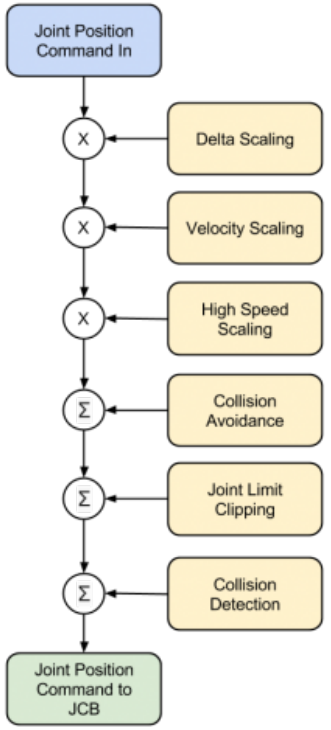
BAXTER's modes of interaction

- Navigators/cuff buttons (on each arm)
- Force sensors (SEA)
- Screen (face) with 1024 x 600 px
- 12 sonar sensors around its head (in a ring)

The 4 control modes of BAXTER

(1) Position mode:

The safest one



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.

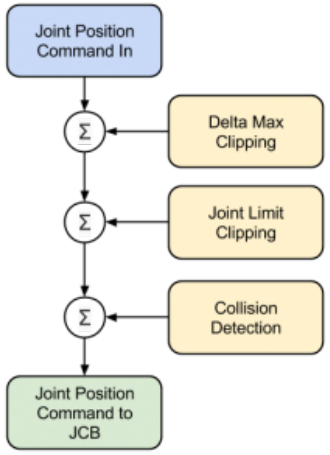
Joint Limit Clipping:
If the joint command is beyond limits, clip the command to respect joint limits.

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

https://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

The 4 control modes of BAXTER

2) Raw Position mode



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

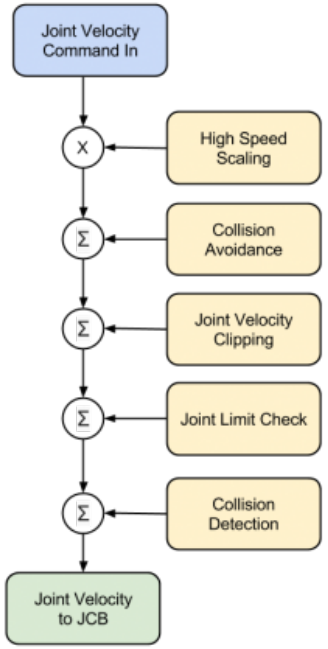
Joint Limit Clipping:
If the joint command is beyond limits, clip the command to respect joint limits.

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

https://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

The 4 control modes of BAXTER

(3) Velocity mode



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.

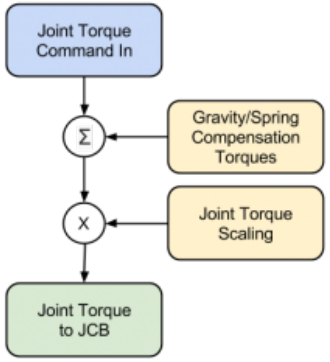
Collision Detection:
If collision (impact) is detected, set position command to hold current compensating for the impact.

https://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

The 4 control modes of BAXTER

(4) Torque mode:

The most dangerous one



Gravity/Spring Compensation:
The joint torque command is applied in addition to the gravity and S1 spring compensation torques.

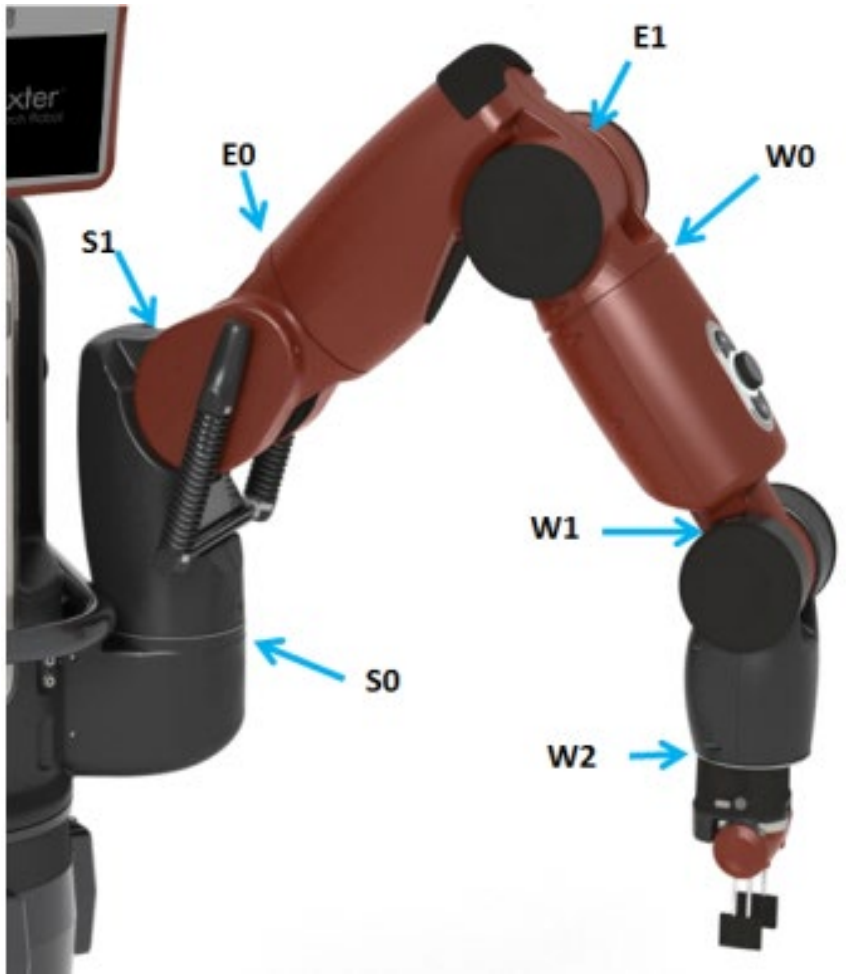
Joint Torque Scaling:
Scales all joint torques if a torque command exceeds the maximum allowable torque for that joint. This scaling ratio is defined as torque_max / torque_command.

https://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

Joint names

Arm:

- **Left**
- **Right**



https://sdk.rethinkrobotics.com/wiki/File:Baxter_arm.png

Basic ROS commands you will need to know:

- *rostopic list*
  - Lists all topics currently available .
- *rostopic echo <topic-name>*
  - Show messages published to a topic.
- *rostopic pub /robot/<topic-name> <msg> -r <rate>*
  - Publishes message (command) to a topic with the specified rate.
  - Example:
    - rostopic pub /robot/limb/left/joint_command baxter_core_msgs/JointCommand
      - "{mode: 1, command: [0.0, 0.0, 0.0, 0.0], names: ['left_w1', 'left_e1', 'left_s0', 'left_s1']}" -r 10

Basic ROS commands you will need to know:

- *rostopic hz /robot/<topic-name>*
  - Show publishing rate of a topic.
- *rostopic type /robot/<topic-name>*
  - Show information about topic's message type.
- *rosmsg show <msg>*
  - Show message description
- *rqt_plot /robot/<topic-name>*
  - Presents the data as a 2D plot.
- *rosrun <package> <executable>*
  - Runs an executable in an arbitrary package without having to give its full path.

Hands-On with BAXTER

Using a VM mashine

Enabling communication between BAXTER and the PC

- Power up the BAXTER robot (white button on robot's back) and your PC.

- Connect to the router on baxter via wifi or cable – make sure that you gets an IP for your VM in the range ”192.168.1.xxx”
  - ssid: ”AAU-BaxterBaxterHome”
  - password: <no password>

- Ping the baxter robot – ping 192.168.1.25 (first from main system, then from the VM system)

- If connected, you can continue

Enabling communication between BAXTER and the PC

- Now, find you IP in the VM – command ”ifconfig” – ip should be 192.168.1.xxx

- Now put this into the ”StartBaxterRobot.sh” gedit ~/ros_ws/StartBaxterRobot.sh

```
24 # Set *Either* your computers ip address or hostname. Please note if using
25 # your_hostname that this must be resolvable to Baxter.
26 your_ip="192.168.1.53"
```

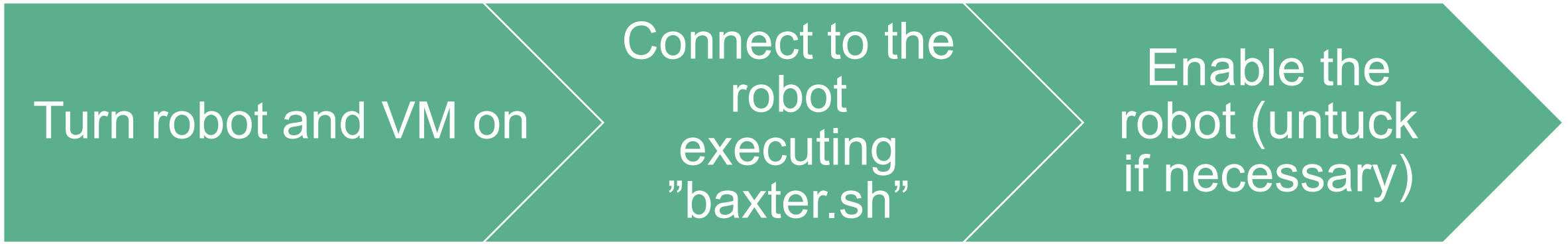
Your IP

Enabling communication between BAXTER and the PC

- Go into the folder which contains ROS environment setup script and execute it:
  - cd ~/ros_ws/
  - ./baxter.sh

- If you want to simulate the robot then
  - ./baxter.sh sim

Enabling communication between BAXTER and the PC



Enabling communication between BAXTER and the PC

- Then test your connection to the master
  - rostopic list
- If it does not work, check the environment variables
  - env | grep ROS

- Enabling the robot directly using the enable_robot script:
  - rosrun baxter_tools enable_robot.py -e
- by untucking the arms:
  - rosrun baxter_tools tuck_arms.py -u

Enabling communication between BAXTER and the PC

- Because BAXTER can only use two cameras simultaneously. It is necessary to choose which ones are enabled:
  - rosrun baxter_tools camera_control.py -l
  - rosrun baxter_tools camera_control.py -c right_hand_camera
  - rosrun baxter_tools camera_control.py -o head_camera -r 1280x800
  - rosrun image_view image_view image:=/cameras/head_camera/image

Simulating Baxter

- On the VM mashine the Simulator are installed

- So you can start from pp. 201 in ”ROS robotics by Example” and you do not need to install any more for simulating baxter.

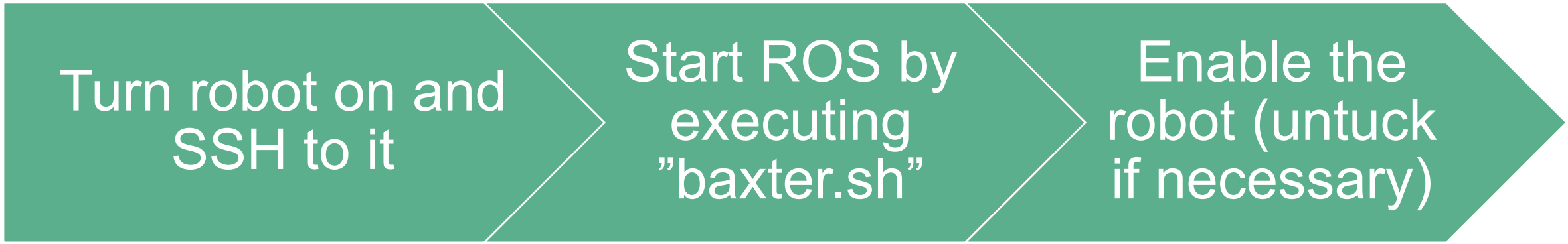
Hands-On with BAXTER

Using a SSH directly to the machine

Enabling communication between BAXTER and the PC

- Power up the BAXTER robot (white button on robot's back) and your PC.
- Download a SSH client (Terminal for Linux/MAC, Putty/CMD/Powershell for Windows)
- Connect to the router on baxter via wifi or cable – make sure that you gets an IP for your machine in the range ”192.168.1.xxx”
  - ssid: ”AAU-BaxterBaxterHome”
  - password: <no password>
- Ping the baxter robot – ping 192.168.1.25 (first from main system)
- If respons, you can continue

Enabling communication between BAXTER and the PC (Via SSH)



Enabling communication between BAXTER and the PC

- Then test your connection to the master
  - rostopic list
- If it does not work, check the environment variables
  - env | grep ROS

- Enabling the robot directly using the enable_robot script:
  - rosrun baxter_tools enable_robot.py -e
- by untucking the arms:
  - rosrun baxter_tools tuck_arms.py -u

ROS Python Basics:

- You can interact with ROS in at least three different ways:
  - Command line
  - C++
  - Python

- I am mainly using Python (rospy) to interact with BAXTER.

ROS Python Basics

Examples of basic Python script with rospy

```
import rospy
rospy.init_node('my_node_name')
while not rospy.is_shutdown():
<do some work>
```

Or

```
import rospy
rospy.init_node('my_node_name')
<... setup callbacks>
rospy.spin()
```


For more details : http://wiki.ros.org/rospy/Overview

ROS Python Basics

BAXTER can be controlled using "pure" Rospy commands as the example below:

```
import rospy
from baxter_core_msgs.msg import JointCommand
pub_joints = rospy.Publisher('/robot/limb/left/joint_command', JointCommand)
rospy.init_node('write_to_ros', anonymous=False)
rate = rospy.Rate(50)
cmd_msg = JointCommand()
cmd_msg.mode = JointCommand.POSITION_MODE
cmd_msg.names = ['left_s0','left_s1']
while not rospy.is_shutdown():
cmd_msg.command = [ 0.1, 1.3 ]
pub_joints.publish(cmd_msg)
rate.sleep()
```

ROS Python Basics

- However, Rethink Robotics has an SDK that provides a set of APIs to simplify the way you write your Python scripts:
  - http://sdk.rethinkrobotics.com/wiki/API_Reference

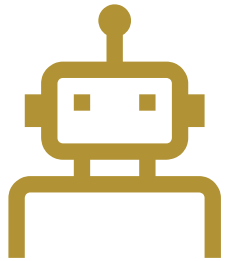
- In the next steps we will use those API calls to control the robot.

Assignments

Assignment 1 : Hallo BAXTER

Follow the instructions on the link below:

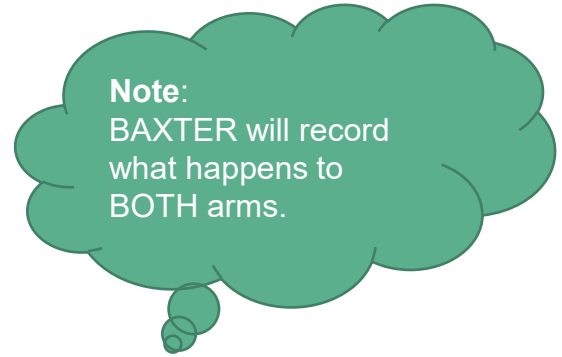
http://sdk.rethinkrobotics.com/wiki/Hello_Baxter



Baxter serie #

[baxter - http://odin.local:11311]

Assignment 2 : Record and Playback



- To record movements and actions:
  - rosrun baxter_example joint_recorder.py -f <filename>

- To play back what was recorded:
  - rosrun baxter_example joint_position_file_playback.py -f <filename>

https://sdk.rethinkrobotics.com/wiki/Joint_Position_Example#Recording_Joint_Positions

https://sdk.rethinkrobotics.com/wiki/Joint_Position_Example#Playback_Recordings

Assignment 3 : Change BAXTER's face

Screen resolution :
1024 x 600

- Choose one face from the link below or use your own file:
  - https://github.com/nfitter/BaxterFaces
- The image displayed on the robot's face can be easily
- changed using the command :
  - rosrun baxter_examples xdisplay_image.py --file=<file>

In Ros_ws there are a file called "face.png"

https://sdk.rethinkrobotics.com/wiki/Display_Image_Example

Assignment 4: Pick-and-place using posion control

- Run the example script as follows :
  - rosrun baxter_examples joint_position_keyboard.py

- Using keyboard control try to pick up the objects and place them in the box, using the gripper.

- https://sdk.rethinkrobotics.com/wiki/Joint_Position_Keyboard_-_Code_Walkthrough

Assignment 5 (adv):
Pick-and-place with the help of inverse kinematics

- First, learn how to use rostopic to verify the current pose:
  - rostopic echo /robot/limb/**<left/right>**/endpoint_state/pose -n 1

- Then download the files ”ik_client.py” and ”ik_client_example.py” from GitHub (code)

- Test them, and then Modify ik_client_example.py according to your needs.

https://sdk.rethinkrobotics.com/wiki/API_Reference#Inverse_Kinematics_Solver_Service

Finishing off

- Tuck the robot's arms:
  - rosrun baxter_tools tuck_arms.py –t
- Switch off BAXTER
- And Smile ☺

Find the material here:
https://github.com/glinvad/AAU-Baxter

Just seach github for **"aaubaxter"**