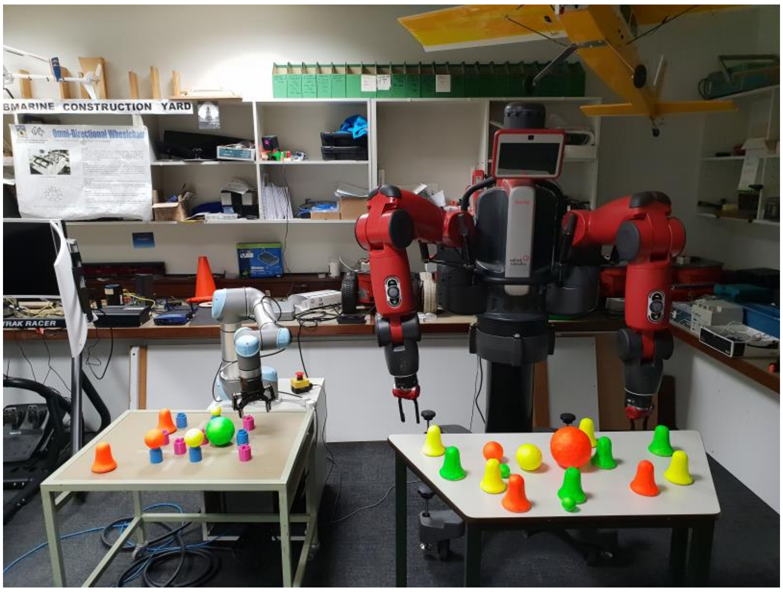
The University of Western Australia
Dept. of Electrical, Electronic and Computer Engineering
Professor Thomas Bräunl

Robotics

GENG5508

Group Project	**Manipulators**	**Weeks 7-12**

Pick and place



Each group will be assigned a robot manipulator (either Baxter or UR5) for this project.

Tasks to complete

Implement a program that allows the manipulator to sort objects of various sizes, shapes, and colours into bins.

- Locate & identify objects of different types (shapes and colours) placed in a pre-defined area.
- Compute world coordinates of objects from the camera coordinates.
- Pick up all objects of a specified type and move them to the corresponding bin.
- Communicate task progress and problem states to the operator.
- Program the manipulator to move smoothly and quickly. This will be judged relative to other groups using the same manipulator.

Bonus tasks:

- Find the bin location automatically (or let the operator teach the location)
- Allow the operator to teach new object types (shapes and colours) interactively.
- Let the manipulator pick up objects placed on a non-level surface.
- Allow the operator to place objects and bins outside the manipulator's initial field of view and have the robot search for objects and bins.

Mid-project assessment (10% of project mark)
To ensure teams are making progress with the project, each team will be asked to demonstrate the following tasks during their **Week 10** consultation session (i.e. the session after the study break):

1. Programmatically move the robot arm from one point to another.
2. Programmatically open and close the gripper.
3. Capture an image from a robot camera and display it on the laptop screen.

Successful demonstration of these tasks will be awarded **0.5 marks each** (out of 15 for the entire project).

Getting started

Baxter:

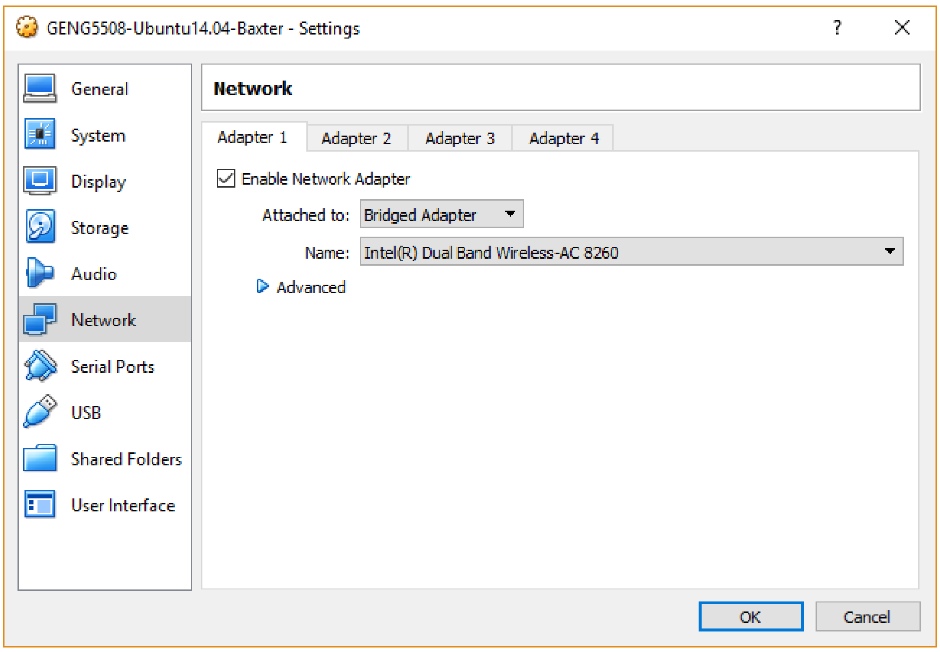
Baxter can be programmed using the Robot Operating System (ROS) framework. To make use of this, a computer running Ubuntu Linux is needed. The easiest way to get started with Ubuntu is by using a virtual machine and the system image provided on the unit webpage. This image contains Ubuntu 14.04 with ROS Indigo installed. Follow the steps below to install the system image:

1. Download VirtualBox at https://www.virtualbox.org/wiki/Downloads
2. Download the image for your manipulator from the unit webpage. http://robotics.ee.uwa.edu.au/courses/robotics/project/general/GENG5508-Ubuntu14.04-Baxter-20180909.ova
3. Import the image into VirtualBox using these instructions: https://www.virtualbox.org/manual/ch01.html#ovf
4. Start the virtual machine and log in:

Username: student
Password: geng5508

Workstation Setup
To connect to Baxter, the following steps must be taken:

1. In VirtualBox, check 'Bridged Network Mode' is set. Start the virtual machine.

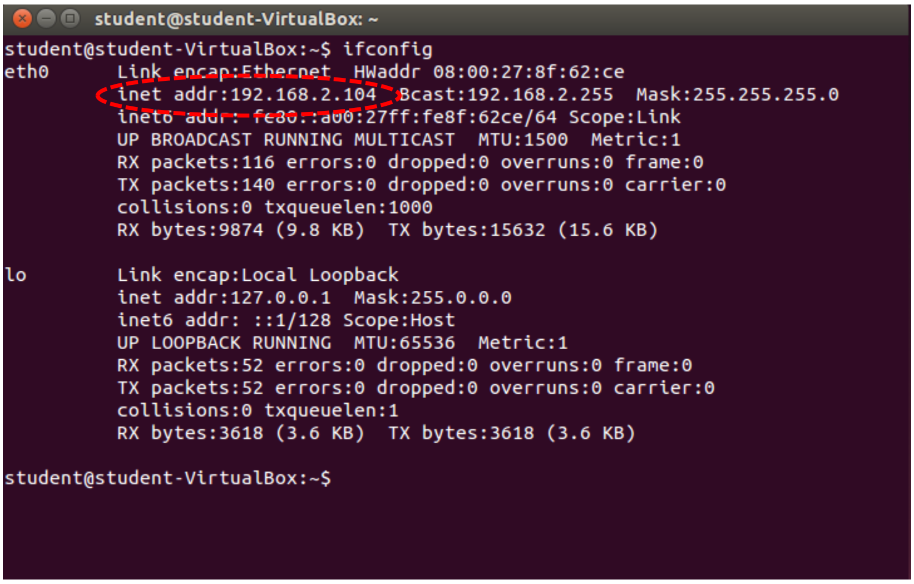


2. Enter the IP address of your computer into the Baxter source script:

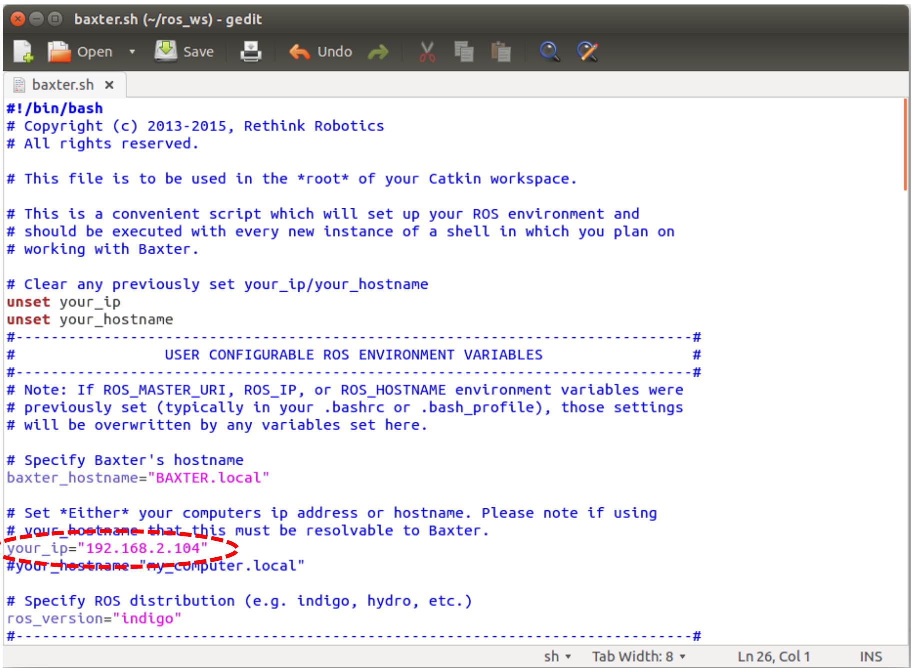
    a. Connect to the Baxter modem:

        SSID: NETGEAR64
        Password: noisytrain086

    b. In Ubuntu, open Terminal and type ‘ifconfig’ and hit Enter. Note the IP address of your computer.



    c. Open the File manager, navigate to ‘home/ros_ws’, and double-click ‘baxter.sh’ to open it in a text editor, such as Gedit. Enter the IP address from the previous step in the ‘your_ip’ field, then save and close the file.



3. Run the source script.

    a. In Terminal, change to the ‘ros_ws’ directory:

        $ cd ros_ws

    b. Run the source script:

        $. baxter.sh

```
student@student-VirtualBox: ~/ros_ws
student@student-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:74:08:7b
          inet addr:10.1.1.105  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe74:87b/64 Scope:Link
          inet6 addr: fd6c:51e6:48c8:0:a00:27ff:fe74:87b/64 Scope:Global
          inet6 addr: fd6c:51e6:48c8:0:70bb:52b6:bf8b:aa0a/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1203 errors:0 dropped:0 overruns:0 frame:0
          TX packets:86 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:75022 (75.0 KB)  TX bytes:15038 (15.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:2805 (2.8 KB)  TX bytes:2805 (2.8 KB)

student@student-VirtualBox:~$ cd ros_ws
student@student-VirtualBox:~/ros_ws$ . baxter.sh
[baxter - http://BAXTER.local:11311] student@student-VirtualBox:~/ros_ws$
```

4. Verify connection to Baxter by typing ‘ping BAXTER.local’ in Terminal.

<u>Baxter Initialisation</u>

The following steps must be performed before Baxter can be used.

1. Complete the workstation setup from the previous section.
2. Enable the robot by typing ‘rosrun baxter_tools enable_robot.py -e’
3. Untuck Baxter’s arms by typing ‘rosrun baxter_tools tuck_arms.py -u’
4. Start a ROS master node in Python:

```
student@student-VirtualBox: ~/ros_ws
student@student-VirtualBox:~$ cd ros_ws
student@student-VirtualBox:~/ros_ws$ . baxter.sh
[baxter - http://BAXTER.local:11311] student@student-VirtualBox:~/ros_ws$ python
Python 2.7.6 (default, Nov 23 2017, 15:49:48)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import rospy
>>> rospy.init_node('my_node')
```

Basic functions

After setting up the workstation and completing the initialisation, Baxter can be controlled. Try typing in the following commands in Python to get started:

Using the grippers:

- Import the baxter_interface module
  `import baxter_interface`
- Create a Python object to control the right gripper
  `right_gripper = baxter_interface.Gripper('right')`
- Calibrate the gripper
  `right_gripper.calibrate()`
- Set the gripper position to 100
  `right_gripper.command_position(100)`

Moving the arms:

- Import the baxter_interface module
  `import baxter_interface`
- Create a Python object to control the right limb
  `right_limb = baxter_interface.Limb('right')`
- Save the current joint angles
  `current_angles = right_limb.joint_angles()`
- Define new joint angles
  `new_angles = current_angles`
  `new_angles['right_e1'] = 2`
- Set the limb position to the new joint angles
  `right_limb.set_joint_positions(new_angles)`

Tidying up

To pack up Baxter, run the tuck_arms tool

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

Resources

- The Baxter Wiki website can be downloaded from the unit page:
  http://robotics.ee.uwa.edu.au/courses/robotics/project/baxter/BaxterWiki_20181109.7z
  To access it, unzip it using 7zip, then open 'index.html'. This contains:
- Rethink Robotics Wiki Homepage: (Home - sdk-wiki)
- API: (Baxter SDK API Documentation)

UR5:
The UR5 manipulator used in this project consists of three components:

- The Universal Robots UR5 arm;
- The Robotiq Wrist Camera; and
- The Robotiq 2F-140 Adaptive Gripper

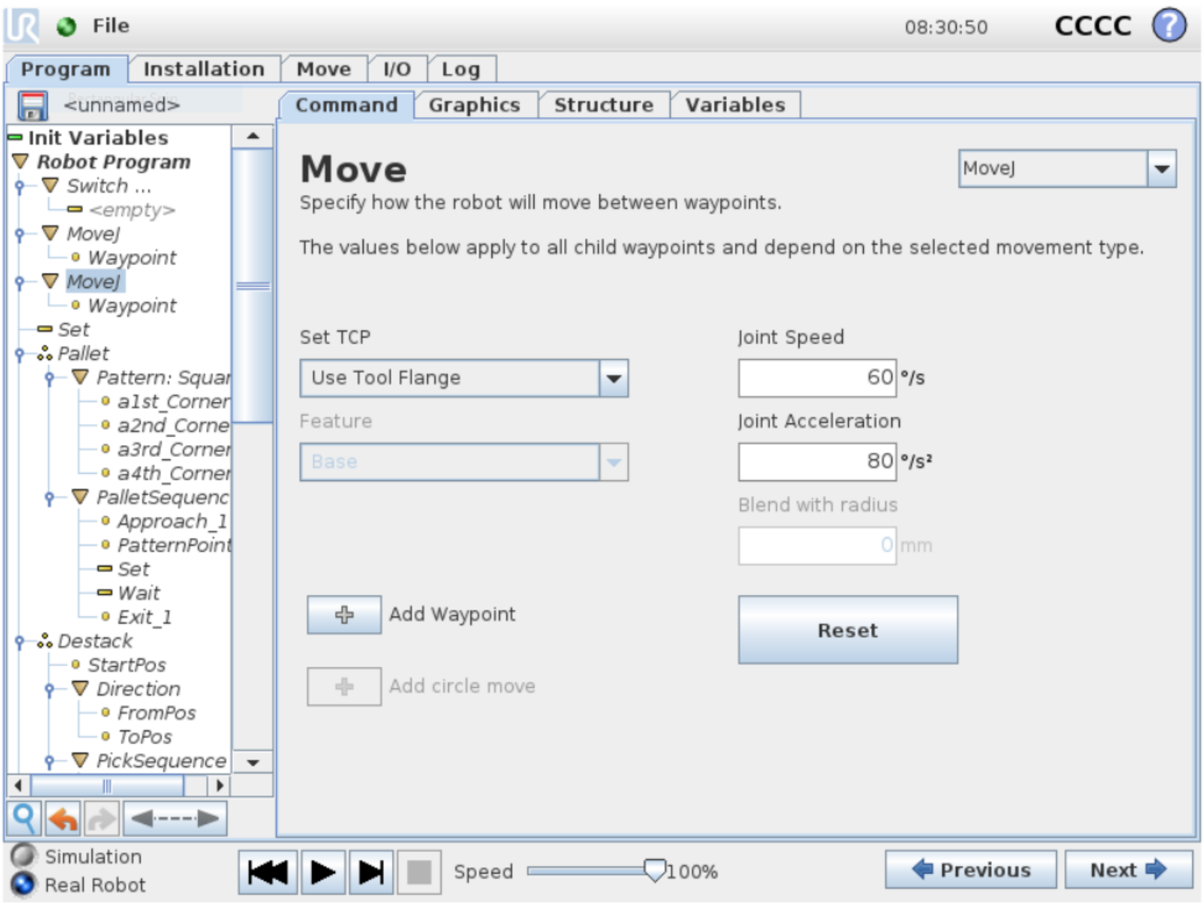
Students are advised to familiarise themselves with the resources referenced below when completing this project.

There are two ways UR5 can be programmed: graphically, using the PolyScope Graphical User Interface (GUI); or programmatically, using URScript. General information on UR5 can be found in the UR5 User Manual:

> https://s3-eu-west-1.amazonaws.com/ur-support-site/27421/UR5_User_Manual_en_Global-3.4.5.pdf

PolyScope GUI programming involves writing the program as a tree structure. Programs must be written using the UR5 Teach Pendant (a touch screen connected to UR5). Commands can be selected using the GUI and inserted into the tree. Information on basic commands can be found in the PolyScope Manual:

> https://s3-eu-west-1.amazonaws.com/ur-support-site/27541/Software_Manual_en_Global-3.4.4.pdf



The source code underlying the PolyScope GUI program is written in URScript. An advantage of programming in URScript is that extended access to the Teach Pendant is not required – programs can be written in a text editor, meaning access to the Teach Pendant is only required when uploading and testing the program. Information on program structure, basic commands, and device interfacing can be found in the URScript Manual:

> https://s3-eu-west-1.amazonaws.com/ur-support-site/28901/scriptManual-3.4.5.pdf

myMove_URscript_4.txt - Notepad

File Edit Format View Help

```
def  Move(x, y, z, rx, ry, rz):
  movel(p[x, y, z, rx, ry, rz], a=1, v=1)
  sleep(1.0)
  x = x + 0.01
  y = y + 0.01
  z = z + 0.01
  movel(p[x, y, z, rx, ry, rz], a=1, v=1)
  sleep(1.0)
end
```

Additional tutorials covering PolyScope GUI and URScript programming can be found here:

> http://www.zacobria.com/universal-robots-knowledge-base-tech-support-forum-hints-tips/

Information on controlling the gripper and camera can be found here in the relevant user manuals:

> https://assets.robotiq.com/production/support_documents/document/Vision_System_PDF_20180725.pdf

> https://assets.robotiq.com/production/support_documents/document/2F-85_2F-140_Instruction_Manual_PDF_20180725.pdf

A third-party library for programming UR5 with Python can be found here:

> https://github.com/SintefManufacturing/python-urx

Robot simulation software is available on marked computers in computer room 2.71:

> https://robodk.com/